Short & Sweet Math Challenges for HP fans #1

*Posted by **Ex-PPC member** on **27 May 2002, 9:59 a.m.***

Hi HP lovers:

Here is a new, tentative 'column' for those HP fans mathematically inclined like myself. If you like math, specially the recreational, funny variety, you're bound to like this 'column'.

If the number of (hopefully positive) responses warrants it, I will post a new math 'challenge' every Monday or so. If not, well, at least I tried ...

This week, a sweet trig challenge in 6 short steps:

1) Take your favourite HP handheld (any will do as long as it can do trigs)

2) Set it to [Degrees] trig mode

3) First, compute: tan(59) + tan(60) + tan(61)

4) Then, compute : tan(59) * tan(60) * tan(61)

5) Explain the result

6) Enlightened by your clever explanation, write a very short program for your HP handheld which will output many other sets of 3 numbers with the same property.

That's all, folks. Best regards.

Re: Short & Sweet Math Challenges for HP fans #1

*Posted by **Pascal** on **27 May 2002, 6:12 p.m.**, in response to Short & Sweet Math Challenges for HP fans #1, posted by Ex-PPC member on 27 May 2002, 9:59 a.m.*

Interesting feature :-)

Obviously there's a symmetry: tan(x-e)+tan(x)+tan(x+e) = tan(x-e)+tan(x)+tan(x+e)

So, if e goes to 0, you get 3*tan(x) = tan(x)^3 <=> tan(x)^2 = sqrt(3) <=> x = atan(sqrt(3)) <=> x = 60 degrees

So for x=60, e can be basically anything as long as you stay within the defined range for tan. Incidentally, this works also for x=-60 deg, x=120 deg, and so on...

Why? Well, I don't know. The "center" points have in common that their 1st derivative is 4 [given by 1/cos^2].

Once these properties are known, writing a program to produce those triplets is easy enough.

But that's only my 5 cents, and I'm already looking forward to a better explanation...

Pascal

Just off the top of my head...

*Posted by **Steve (Australia)** on **27 May 2002, 10:20 p.m.**, in response to Short & Sweet Math Challenges for HP fans #1, posted by Ex-PPC member on 27 May 2002, 9:59 a.m.*

excuse any errors, I haven't got a calculator here :-)

tan(a +/- b) = (tan a +/- tan b)/(1 -/+ tan a tan b)

tan(60 +/- x) - (sqrt 3 +/- tan x)/(1 -/+ sqrt 3 tan x)

tan(60 - x) + tan 60 + tan (60 + x) = (sqrt 3 - tan x)/(1 + sqrt3 tanx) + sqrt3 + (sqrt 3 + tanx)/(1 - sqrt3 tanx)

= sqrt 3 + ((sqrt 3 - tanx)(1 - sqrt3 tanx) + (sqrt 3 + tanx)(1 + sqrt3 tanx))/(1 + sqrt3 tanx)(1 - sqrt 3 tanx)

= sqrt 3 + (sqrt3 - tanx - 3tanx + sqrt3tan^2x + sqrt3 + tanx + 3tanx + sqrt3tan^2x)/(1 - 3tan^2x)

= sqrt3 + (2sqrt3 + 2sqrt3tan^2x)/(1 - 3tan^2x)

= [sqrt3(1 - 3tan^2x) + 2sqrt3 + 2sqrt3tan^2x]/(1 - 3tan^2x)

= [sqrt3 - 3sqrt3tan^2x + 2sqrt3 + 2sqrt3tan^2x]/(1 - 3tan^2x)

= [3sqrt3 - sqrt3tan^2x]/(1 - 3tan^2x)

= sqrt3(3 - tan^2x)/(1 - 3tan^2x)

tan(60-x)Tan(60)tan(60+x)

= (sqrt 3 - tan x)/(1 + sqrt3 tanx) . sqrt3 . (sqrt 3 + tanx)/(1 - sqrt3 tanx)

= [sqrt3 . (sqrt3 + tanx) . (sqrt3 - tanx)] / [(1 + sqrt3 tanx) . (1 - sqrt3 tanx)]

= sqrt3(3 - tan^2x)/(1 - 3tan^2x)

so they're equal!

so subject to the usual restrictions the +/- 1 used in the example can be any real number.

Is the HP48/49 up to the task ?

*Posted by **Ex-PPC member** on **28 May 2002, 7:15 p.m.**, in response to Just off the top of my head..., posted by Steve (Australia) on 27 May 2002, 10:20 p.m.*

Congratulations to those keen people that saw that the required condition is:

a + b + c = 180 (degrees, or 2Pi radians)

Now the question is: can the advanced symbolic math capabilities of the HP48/49 models help with the demonstration ?

Basically, you need to transform, rearrange, and generally symbolically manipulate this identity to get one side given the other (or to reduce both sides to a common intermediate expression):

tan(a)+tan(b)+tan(180-a-b) = tan(a)*tan(b)*tan(180-a-b)

(or 2Pi instead of 180, if using radians). This is a much harder nut than the usual run-of-the-mill trigonometric identities frequently seen as didactic examples.

Can your symbolic-math-capable HP handheld help with it ?

Re: Is the HP48/49 up to the task ?

*Posted by **Mike Markowski** on **7 June 2002, 10:56 a.m.**, in response to Is the HP48/49 up to the task ?, posted by Ex-PPC member on 28 May 2002, 7:15 p.m.*

The hp 49g can do almost all of it. First, you the wielder of the blue box must realize that a+b+c=180 degrees. (I wish I could format this post.) Next, in the equation writer I entered: TAN(X) + TAN(Y) + TAN(pi - (X+Y)) and then hit TEXPAND from the TRIG menu. Then I hit FACTOR from the ALG menu. At that point the display read TAN(X) TAN(Y) (TAN(X) + TAN(Y)) / (TAN(X) TAN(Y) - 1) and it was up to me to realize that the final factor was really TAN(pi - (X+Y)). So the brain was needed for a little bit of it. :-) -- Mike

Re: Just off the top of my head...

*Posted by **John Ioannidis** on **29 May 2002, 8:35 a.m.**, in response to Just off the top of my head..., posted by Steve (Australia) on 27 May 2002, 10:20 p.m.*

It's much easier than that: the property holds for any three angles A, B, C such that A+B+C = 2*pi. Then:

tanA+tanB+tanC = tanA + tanB + tan(2*pi - A - B) = tanA + tanB-tan(A+B)

tanA + tanB = tanA + tanB - ------------- (1-tanA*tanB)

(tanA + tanB)*(-tanA*tanB) = ---------------------------- (1-tanA*tanB)

tanA + tanB = tanA * tanB * (- -------------) (1-tanA*tanB)

= tanA * tanB * (- tan(A+B))

= tanA * tanB * tan(2*pi - A - B)

= tanA * tanB * tanC

QED

Re: Just off the top of my head...

*Posted by **John Ioannidis** on **29 May 2002, 4:07 p.m.**, in response to Re: Just off the top of my head..., posted by John Ioannidis on 29 May 2002, 8:35 a.m.*

Bloody 'ell! I forgot that things get reformatted here. OH well, someone else has effectively the same proof in this thread.

One of my favorite trigonometric (well, geometric really) properties is that the area of a triange is sqrt(t*(t-a)*(t-b)*(t-c)), where a, b, and c are the lengths of the sides, and t=(a+b+c)/2

Thanks to both and yes, but ...

Posted by ***Ex-PPC member*** *on* ***28 May 2002, 9:24 a.m.****, in response to Short & Sweet Math Challenges for HP fans #1, posted by Ex-PPC member on 27 May 2002, 9:59 a.m.*

... just a little hint:

The triplet (37, 73, 70) is also a solution:

tan(37) + tan(73) + tan(70) = tan(37) * tan(73) * tan(70)

as you can easily verify with your favorite HP calc, yet, it isn't of the form (60-x, 60, 60+x) as you can easily observe ...

Perhaps it will be enlightening (if slow) to give a chance to empirical methods. After all, most HP calcs do not do symbolic math. You could for instance run a version for your HP of this pseudo-code:

```
set degrees trig mode
for a = 0 to 360
  for b = 0 to 360
    for c = 0 to 360
      if tan(a)+tan(b)+tan(c)=tan(a)*tan(b)*tan(c) then print "Solution:",a,b,c
    next c
  next b
next a
```

and see if you can deduct anything from the solution triplets printed. Once enlightened, you can use your wonderful 48/49 to do the symbolic math needed to prove your clever conjecture.

[of course, the above program would need to be optimized to run in acceptable times, and to cater for rounding errors in the computation before testing for equality, not to mention range errors]

Re: Thanks to both and yes, but ...

Posted by ***thibaut.be*** *on* ***28 May 2002, 10:00 a.m.****, in response to Thanks to both and yes, but ..., posted by Ex-PPC member on 28 May 2002, 9:24 a.m.*

It seems than when the sum of a+b+c = 180° (or pi), the product and the sums of the tans are equal... I'm currently running this routine (not on a HP calc, because 360^3 loops 46.656.000 loops are too much, and I'll see if I can deduct something...

But why, I don't know. I've had a look at my trigonometric formulaes, but I haven't found yet something that rang me a bell...

Re: Thanks to both and yes, but ...

*Posted by **Dave Shaffer** on **28 May 2002, 12:08 p.m.**, in response to Re: Thanks to both and yes, but ..., posted by thibaut.be on 28 May 2002, 10:00 a.m.*

Thibaut seems to be on the right track.

A simple BASIC program indicates that when the sum of the angles equals 180 degrees, the product of the tangents equals the sum of the tangents. I haven't done the expansions yet, but I suspect the method used by Steve, applied (recursuvely) to tan(a + (b+c)), with the condition that a+b+c = 180, will prove fruitful.

By the way, when writing your programs (HP or otherwise) to check sum(tangents) = prod(tangents), don't forget that the proper check is to determine whether the (absolute value of the) DIFFERENCE is within some tolerance (that depends on the number of significant figures in your computer). 1e-5 or 1e-6 (to use old-fashioned (?) FORTRAN notation) is probably good enough. This is something you learn on day 2 or 3 of a programming techniques course.

Re: Thanks to both and yes, but ...

*Posted by **thibaut.be** on **28 May 2002, 2:18 p.m.**, in response to Re: Thanks to both and yes, but ..., posted by Dave Shaffer on 28 May 2002, 12:08 p.m.*

I forgot to mention than none of a,b, or c must be = 0 of course.

But I'm still expecting the maths demonstration

Re: Thanks to both and yes, but ...

*Posted by **Steve (Australia)** on **28 May 2002, 7:44 p.m.**, in response to Re: Thanks to both and yes, but ..., posted by Dave Shaffer on 28 May 2002, 12:08 p.m.*

Ah! If the rule is that the sum of the angles must be 180 then I suspect you should be looking at some if the identities involving triangles :-)

(I'm rather too busy at the moment)

Re: Thanks to both and yes, but ...

Posted by ***Steve (Australia)*** *on* ***28 May 2002, 8:01 p.m.****, in response to Thanks to both and yes, but ..., posted by Ex-PPC member on 28 May 2002, 9:24 a.m.*

since both addition and multiplication are commutative, and because the solution a, b, c is the same as c, a, b and c, b, a, ... you can halve your execution times as follows:

```
for a = 0 to 360

for b = a to 360 <---------- this

for c = b to 360 <---------- and this

if tan(a)+tan(b)+tan(c)=tan(a)*tan(b)*tan(c) then print "Solution:",a,b,c

next c

next b

next a
```

and because all possible values of tan can be found in the interval -90 to 90, we can reduce the number of iterations (as well as the nasty discontinuities) by

```
for a = -89 to 89

for b = a to 89

for c = b to 89

if tan(a)+tan(b)+tan(c)=tan(a)*tan(b)*tan(c) then print "Solution:",a,b,c

next c

next b

next a
```

if we're only interested in the domain 0 to 360, then we need to report +ve a, b, c as

a or 180 - a

and negative a, b, c as

360 + a or 180 + a

then find a general solution by inspection :-)

then you need to convince yourself that there are *not* solutions at the discontinuities ;-)

Obvious correction

*Posted by **Steve (Australia)** on **28 May 2002, 8:26 p.m.**, in response to Re: Thanks to both and yes, but ..., posted by Steve (Australia) on 28 May 2002, 8:01 p.m.*

when I said

> a or 180 - a

> and negative a, b, c as

> 360 + a or 180 + a

I obviously didn't mean that :-)

a or 180 + a

and negative a, b, c as

360 + a or 180 + a

and I have found empirically that a + b + c must be a multiple of 180

I would tend to suspect that 90, 1, 89 would be a solution, but would anyone care to comment?

By the same token, 90, 90, 90 would not be a solution.

Hmmmm.....

oh, and I'd write a pseudo code solution for anyone wanting to investigate this in this manner:

```
  i, j, k : integer
  a, b, c : double

  epsilon = 1e-9
begin
  for i := -89 to 89 do
    a := tan(i*pi/180)

    for j := i to 89 do
      b := tan(j*pi/180)

      for k := j to 89 do
        c := tan(k*pi/180)

        if abs((a+b+c) - (a*b*c)) < epsilon then
          solution is i, j, k (sum is i+j+k)
        endif
      next
    next
  next
end;
```

And the answer is ...

*Posted by **Dave Shaffer** on **28 May 2002, 9:45 p.m.**, in response to Obvious correction, posted by Steve (Australia) on 28 May 2002, 8:26 p.m.*

With the a+b+c = 180 degrees condition, things are trivial.

Recall Steve's initial fact:

tan(a+b) = (tan a + tan b)/(1 - (tan a)*(tan b))

which can be looked up in your favorite book of trig identities. Note, also, that tan(-a) = -tan a .

Then, with the 180 degree condition, we need to determine whether

tan a + tan b + tan(180-a-b) = (tan a)*(tan b)*tan(180-a-b)

Note, also, that tan(180-x) = -tan x (which can be derived from the sum formula above, realizing that tan 180 = 0).

Then we get tan(180-a-b) = -tan(a+b)

This relation greatly simplifies our task!

To keep from typing "tan" all over the place, I let tan a = a, tan b = b, and tan c = c everywhere below. Since only tangents appear, there is no loss of meaning.

Then, our expression to check becomes (with the 180 degree identity included, which puts in some minus signs):

a + b - tan(a+b) =? -a * b * tan(a+b)

where tan(a+b) has its normal meaning (i.e. NOT tan(tan a + tan b)) !!

Now, expand tan(a+b) to get

a + b - [(a + b)/(1-ab)] =? -a * b * [(a + b)/(1-ab)]

where a, b, c now all mean their respective tangents.

Multiply this expression through by the (1-ab) denominator to get

(1-ab)(a+b) - a - b =? -a * b * (a+b)

This becomes

a + b - aab - abb -a - b =? -aab - abb

where I have elected to write aa (and bb) instead of a^2 (a-squared) for a times a.

The a and b terms cancel on the left side, leaving us with an identity: -aab -abb = -aab -abb

So, indeed, if the sum of the angles equals 180 degrees, the sum and product of the three tangent terms ARE EQUAL.

QED

Next challenge?!

I think we're off on a tangent

*Posted by **Steve (Australia)** on **29 May 2002, 8:10 p.m.**, in response to And the answer is ..., posted by Dave Shaffer on 28 May 2002, 9:45 p.m.*

groan

```
> So, indeed, if the sum of the angles equals 180 degrees,
> the sum and product of the three tangent terms ARE EQUAL.
>
> QED
```

If the sum of the angles is a multiple of 180

But what of the case where one of the angles is 90 or 270 (-90, 450, ...)???

You will be asking for a tan at one of the points where the result is undefined.

Is the relationship still valid at these points? Are the infinities equal?

Final remarks

Posted by ***Ex-PPC member*** *on* ***30 May 2002, 12:33 p.m.****, in response to Short & Sweet Math Challenges for HP fans #1, posted by Ex-PPC member on 27 May 2002, 9:59 a.m.*

Thanks to all those people that posted and/or were interested in this thread, I think the issue has been made perfectly clear, just a few final remarks.

First, the original challenge could have been presented like this:

"What condition must a,b,c meet for us to have:

Ln(a + b + c) = Ln(a) + Ln(b) + Ln(c) ?"

where, weirdly enough, the log of the sum is equal to the sum of the logs !. The required condition can be expressed of course, as

arctan(a) + arctan(b) + arctan(c) = Pi

which is quite astonishing at first, making a seemingly logarithmic expression depend on a trigonometric identity involving the number Pi.

Second, as a corollary of this fact, we have the beautiful identity:

Pi = arctan(1) + arctan(2) + arctan(3)

which is just a particular case. You just need to realize that 1 + 2 + 3 = 1 * 2 * 3 (= 6) to demonstrate it.

Third, I would love to know if the symbolic math capabilities of the HP48/49 can cope or help with this kind of unusual trigonometric identities. Can some dedicated and loving HP49 user tell us ?

Next S & S Math Challenge soon, stay tuned ! :-)

Re: Final remarks

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***1 June 2002, 10:57 a.m.****, in response to Final remarks, posted by Ex-PPC member on 30 May 2002, 12:33 p.m.*

Thank you, Mr. Ex-PPC (would you ever reveal your identity?), for an enlightening challenge and the "bonus track" with the final remarks!!

Please keep such challenges coming!!

Short & Sweet Math Challenges for HP fans #2

Posted by ***Ex-PPC member*** *on* ***1 June 2002, 9:27 p.m.***

Thanks to all of you who were interested in the very first S&SMC, posted to this forum a week or so ago. Its thread had a total of 17 messages posted (including 3 by myself), so I take this to mean you like the 'section'. Special thanks go to Mr. Andres C. Rodrigues for his enthusiastic support and highly encouraging words. To honour your request, here's a new challenge for all of you and your favourite HP handhelds.

Last week's challenge (the one about the tangents), was mostly theoretical in nature, with just a little programming being useful to empirically try and discover the underlying relationship (a+b+c=180). Once postulated, said relationship could be demonstrated using symbolic math, perhaps with the help of some able HP (such as an Hp48 or HP49), though unfortunately nobody commented on whether those advanced models could cope with it and how.

All in all, it was a rather theoretical (if interesting) challenge, so in order to compensate somewhat, this week's challenge, #2, is much more empirical and will have you and your HPs working hard in order to succeed. It goes like this:

- 1) Take your favourite HP calc which can do matrices, either using fast built-in capabilities or else suitable programs for the task at hand. Specifically, it must be capable of storing a 3x3 or 4x4 matrix and compute its determinant. (Models with that capability built-in in fast microcode include: HP-15C, HP-41C + Advantage ROM, HP-71 + Math ROM, HP-42S, and HP-48/49 models, among several others. Most models can also be easily programmed to compute determinants up to 3x3 at least, so you've got ample choice)
- 2) Now, you must define a 3x3 matrix called A, and fill it up with the integers from 1 to 9, in any order you like as long as none of them are repeated. For instance, you can have your matrix like this:
- `          |  6   1   8  |`
- `      A = |  7   5   3  |`
- `          |  2   9   4  |`

  and its determinant is det(A) = 360
- 3) The challenge is: find the arrangements of the integers from 1 to 9 which:

  1. make det(A) the minimum possible non-zero value.
  2. make det(A) the maximum possible value.

  We are only interested on absolute values of the determinant, regardless of the sign. The solutions are not unique, of course.
- 4) As you can see, there are fact(9) = 362880 possible arrangements of the integers 1,2,...,9 without repetitions, so you'll have to make good use of your ingenuity and your HP's capabilities if you intend to avoid very long computation times.

- 5) If you succeed in the 3x3 case, try your might with the 4x4 case, i.e: find the arrangements of the integers from 1 to 16 which make the determinant of the 4x4 matrix maximum or minimum (but non-zero). The number of possible arrangements is now fact(16), an impossibly large number to use brute force. Some finesse is indeed required. For instance, can you find an arrangement which makes det(A) = 1 ? Or = 4000 ? or = 4444 ? Or > 40000 ? Which is more, can you theoretically demonstrate what are the maximum/minimum values of the determinant for an NxN matrix ?

A hard nut to crack, but thanks to our HP's quality, it can be done ! Give it a chance, refrain from the temptation to use a pc, and let us all see how you and your HP cope with this challenge !

Re: Short & Sweet Math Challenges for HP fans #2

Posted by ***John Ioannidis*** *on* ***2 June 2002, 5:24 p.m.****, in response to Short & Sweet Math Challenges for HP fans #2, posted by Ex-PPC member on 1 June 2002, 9:27 p.m.*

Several observations:

1. The maximum and the mimimum solutions have the same absolute value (you get one by exchanging two rows or two columns of the other). Hence, we are only going to consider as solution the absolute value of the determinant.
2. The lowest solution is obviously 0, one of which is

```
3. | 1 2 3 |
4. | 4 5 6 |
   | 7 8 9 |
```

5. Due to symmetry, the number of solutions for each possible value is a multiple of 72 (exchanging rows or exchanging columns does not change the absolute value of the determinant; there are 3 ways to arrange the rows, and another three to arrange the columns. Rotating or transposing does not change the value either, so there is another factor of 8 there; multiply and you get 72).
6. Again due to symmetry, the top left element can be considered to be 1; this leaves 8! instead of 9! permutations to consider, a much more tractable number on a calculator.
7. The really hard part is generating the permutations; it gets harder on HP calculators because you can't use a recursive algorithm and have to unwind the recursion yourself.

Now for some solutions:

1. The highest value is 412. Here is the first solution

```
2. | 1 4 8 |
3. | 5 9 3 |
   | 7 2 6 |
```

   All the solutions giving 412 are the 72 permutations of this matrix.

4. There are 2736 zero solutions.
5. The most common solution is 45: there are 3456 of those.

Re: Short & Sweet Math Challenges for HP fans #2

*Posted by **Ex-PPC member** on **5 June 2002, 10:15 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #2, posted by John Ioannidis on 2 June 2002, 5:24 p.m.*

Hi Mr. Ioannidis,

It seems that this week's challenge hasn't catch the fancy of math-lover members of this forum, as the only message posted so far has been yours. Thanks for it, but I feel some comments are in order:

Mr. Ioannidis: *"The maximum and the mimimum solutions have the same absolute value (you get one by exchanging two rows or two columns of the other). Hence, we are only going to consider as solution the absolute value of the determinant."*

That's exactly what I said in my description of the challenge: "We are only interested on absolute values of the determinant, regardless of the sign."

Mr. Ioannidis: *"The lowest solution is obviously 0"*

Wrong. In my description of the challenge you can read: "find the arrangements of the integers from 1 to 9 which: 1.make det(A) the minimum possible non-zero value.". Observe that I explicitly requested *non-zero* value. With that condition in mind, the lowest solution cannot be 0, obviously.

Mr. Ioannidis: *"The really hard part is generating the permutations; it gets harder on HP calculators because you can't use a recursive algorithm and have to unwind the recursion yourself."*

Wrong again. At least one model does allow recursion: HP-71B BASIC language allows subprograms and user-defined functions (single-line and multi-line) to call themselves, so permitting recursion naturally.

Mr. Ioannidis: *"Now for some solutions: The highest value is 412."*

Correct. What about the minimum *non-zero* absolute value ?

Mr. Ioannidis: *"There are 2736 zero solutions. 3.The most common solution is 45: there are 3456 of those."*

While this is of course correct, remember I pleaded: " Give it [your HP calculator] a chance, refrain from the temptation to use a pc, and let us all see how you and your HP cope with this challenge !"

While I'm grateful to you for your interest in the challenge and your results, it seems obvious to me that you ignored my plea and did use your PC or computer, not your HP calculator. While this is fun and interesting, you should bear in mind that these challenges are tailored to levels of difficulty affordable for an HP calculator, and are not intended to be solved using a computer, far too easy for that, no challenge at all. No clever techniques to overcome the difficulties either, a brute-force approach suffices.

Let's hope that, before the week elapses, some other kind contributors of this forum will shed some light on the solutions for the minimum 3x3 determinant, and the maximum and minimum 4x4 determinant. Seeing some code for any HP calculator would be great ! And finding a theoretical expression for arbitrary NxN determinants would be eidetic.

So I repeat: put aside you fancy PC at 2 Ghz and let your 71B, your 42S or your 49 do the job, Ok ? :-)

Re: Short & Sweet Math Challenges for HP fans #2

*Posted by **Thibaut.be** on **6 June 2002, 4:04 a.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #2, posted by Ex-PPC member on 5 June 2002, 10:15 p.m.*

Well, I have to admit that in order to apply to this challenge I had to go back to my old maths books.

I love those quizzes, but I propose that you lower a bit the level of the questions, I guess you'd get more answers...

Re: Short & Sweet Math Challenges for HP fans #2

*Posted by **Chris Randle (Lincoln - UK)** on **6 June 2002, 8:53 a.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #2, posted by Ex-PPC member on 5 June 2002, 10:15 p.m.*

At least one model does allow recursion: HP-71B BASIC language allows subprograms and user-defined functions (single-line and multi-line) to call themselves, so permitting recursion naturally

All RPL machines can have programs that call themselves too. If you have a program called FAC to calculate a factorial, then:

```
<< -> n << IF n 1 == THEN 1 ELSE n 1 - FAC n * END >> >>
```

works quite nicely with FAC(n) calling FAC(n-1) * n, etc. Tried it on a 28S. I know the "IF n 1 ==" can be replaced by "IF n", but I thought the intent was clearer.

I'm going to have a go at your puzzle now, instead of doing the work I'm paid for ;-) I love puzzles like these for little machines. Thanks for posting them.

Recursion in HP calcs

*Posted by **Andrés C. Rodríguez (Argentina)** on **6 June 2002, 11:38 a.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #2, posted by Ex-PPC member on 5 June 2002, 10:15 p.m.*

I would just say that the HP 41 programming model (hence including the HP 41C/CV/CX and HP 42S calculators) allow for recursion at the user programs level.

Re: Recursion in HP calcs

*Posted by **John K. (US)** on **6 June 2002, 8:19 p.m.**, in response to Recursion in HP calcs, posted by Andrés C. Rodríguez (Argentina) on 6 June 2002, 11:38 a.m.*

[T]the HP 41 programming model [...] allow for recursion at the user programs level.

Unfortunately, due to the limitations one calling chain depth, this ability is somewhat limited. One of the programming projects I was working on many, many moons ago was a routine to copy out the return address registers in the 41 to the main storage registers and back again to seemlessly allow for longer calling chains. I don't think I ever got it working quite right and, since it was (obviously) filled with synthetic commands, it won't work on a 42S. Still, it might be interesting to see if I still have a copy of it somewhere...

Re: Recursion in HP calcs

*Posted by **Andrés C. Rodríguez (Argentina)** on **7 June 2002, 7:23 a.m.**, in response to Re: Recursion in HP calcs, posted by John K. (US) on 6 June 2002, 8:19 p.m.*

You are right, I should have said that "...the 41 programming model allow for limited recursion"

Re: Recursion in HP calcs -- Pseudocode

*Posted by **Paul Brogger** on **7 June 2002, 6:48 p.m.**, in response to Re: Recursion in HP calcs, posted by John K. (US) on 6 June 2002, 8:19 p.m.*

Here's how I think it would work.

.

.

Scenario:

. Main process named “MP”

. . calls recursive subprocess to figure something.

,

Recursive subprocess named “RSP”

. Figures something in two blocks of code (“RSP_a” & “RSP_b”),

. . with possible recursive self-call in between them.

. Needs to maintain two local variables, “x” & “y” across recursive calls.

,

Logic:

```
. process MP
. . allocate recursive call/return stack
. . initialize stack pointer to 0
. . [ do some work ]
. . initialize x = 0
. . initialize y = 0
. . set caller = “MP” [could be a numeric code, whatever]
. . GSB RSP_a
. . [ do some more work ]
. . clean up & exit.
. . endprocess MP
.
process RSP_a
. [use x & y]
. push x
. push y
. push caller
. IF recursive descent indicated THEN
. . set caller = “RSP_a”
. . GTO RSP_a
. ELSE
. . GTO RSP_b
. endprocess RSP_a
.
process RSP_b
. pop caller
. pop y
. pop x
. [ use x & y ]
. IF caller = “RSP_a” THEN
. . GTO RSP_b
. ELSE
. . RTN
. endprocess RSP_b
.
```

"push" & "pop", of course, would utilize an array as a stack to save values and caller codes.

I'll have to try to implement a general solution to, say, the "Tower of Hanoi problem" on my 42s this weekend . . .

Re: Recursion in HP calcs

*Posted by **Paul Brogger** on **7 June 2002, 12:56 p.m.**, in response to Recursion in HP calcs, posted by Andrés C. Rodríguez (Argentina) on 6 June 2002, 11:38 a.m.*

If one kept one's own "call/return stack" in user memory in the form of codes indicating what was called and where to return, couldn't one simply use GTO rather than XEQ/RTN (or equivalents)? That should extend the depth of the call/return stack greatly. (Especially in, say, a 32K 42s or a 48/49.)

Re: Recursion in HP calcs

*Posted by **thibaut.be** on **7 June 2002, 2:09 p.m.**, in response to Re: Recursion in HP calcs, posted by Paul Brogger on 7 June 2002, 12:56 p.m.*

XEQ <lbl> or GTO <lbl> are exactly the same, aren't they ?

Re: Recursion in HP Calcs

*Posted by **Paul Brogger** on **7 June 2002, 2:26 p.m.**, in response to Re: Recursion in HP calcs, posted by thibaut.be on 7 June 2002, 2:09 p.m.*

Details, details!

Actually, that may betray how little I actually am engaged now in *programming* the things.

On my -32s (and presumably the -32sii) I believe that XEQ means "GOSUB" (call with return) and GTO means "GOTO" (go there without expectation of return). If I've got it right, the first enters a return address on the internal return stack, while the latter doesn't.

My HP-48G doesn't seem to have a GOTO, but perhaps I haven't looked closely enough. Regardless, one or more recursive routines could be broken into segments and sequenced through codes passed to special "Call" and "Return" subroutines. (Without a GOTO, it would just be a little more work.)

Of course, any variables whose values need to be maintained across such a recursive call would need to be maintained by the programmer on a stack as well. As with *any* actual implementation, there would be an real (if not a practical) limitation on just how far the process could recursively descend before the end of memory is reached.

I imagine that general-purpose "Recursive Call" and "Recursive Return" subroutines could be written for each calculator, with some documentation detailing how to use them, and what other requirements there may be of the programming and storage for the actual routines being called. (I wouldn't bother with the -32s -- not enough memory. But the -41, -42S & beyond should be capable of supporting such.)

In fact, I don't know what the limit on subroutine descent is in the 49/49 models. I doubt it's anywhere near as limited as it is on the earlier machines. But I don't have my manual handy . . .

Re: Recursion in HP Calcs

Posted by ***thibaut.be*** *on* ***7 June 2002, 4:56 p.m.****, in response to Re: Recursion in HP Calcs, posted by Paul Brogger on 7 June 2002, 2:26 p.m.*

Well, on the 41C the GSB function is well present. Now I have a doubt. The difference between GTO and GSB is, as most of us know I guess, that the RTN function steps to the next instruction after GSB.

I will check that on my 41, but I think the XEQ function works like the GSB in that sense...

Re: Recursion in HP Calcs

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***7 June 2002, 5:05 p.m.****, in response to Re: Recursion in HP Calcs, posted by thibaut.be on 7 June 2002, 4:56 p.m.*

In fact, XEQ as used in the HP41/42 replaced the GSB mnemonic used in previous models. The discussion about recursion has to do with the "pending returns" depth of an internal "return addresses" stack (nothing to do with the RPN stack, of course). The HP41 had 6 levels of pending returns, the HP42 increased it to 8; in any case this limits the "recursionability" of these models.

So your statement can be enhanced as "the RTN function steps to the next instruction after GSB" [as long as there are no more than "n" pending returns].

Re: GSB Considered Harmful

Posted by ***Paul Brogger*** *on* ***7 June 2002, 5:52 p.m.****, in response to Re: Recursion in HP Calcs, posted by Andrés C. Rodríguez (Argentina) on 7 June 2002, 5:05 p.m.*

Right. We're not talking about the RPN stack at all.

But, my point is, if you *AVOID* using GSB/RTN (or whatever) to call subroutines, you aren't bound by the limited subroutine call depth supported by the calculator's firmware.

To avoid GSB/RTN, one would have to *simulate* GSB/RTN functionality using application logic that is based on GTO (and probably utilizing arrays). This logic would have to save return "addresses" and local variable values on a stack, pushing the values at the time of pseudo-GSB, and popping those values as part of a pseudo-RTN.

So, given some clever programming and enough memory, you needn't be limited by the depth calculator's internal return stack.

Return addresses

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***7 June 2002, 6:11 p.m.****, in response to Re: GSB Considered Harmful, posted by Paul Brogger on 7 June 2002, 5:52 p.m.*

While we may think about keeping our own management of return addresses, it should be noted that you cannot return to an arbitrary program line. You can simulate a return by means of a GTO to an existent label (somehow identified by your "return handling" extensions), or go to a PHYSICAL address (which is most harmful, indeed!) using synthetic programming and a lot of discipline.

While I was the one who stated that HP41/42 could operate recursively, I am also among the first to admit such feature is very limited for general recursion.

Re: Return addresses

*Posted by **John K. (US)** on **7 June 2002, 8:58 p.m.**, in response to Return addresses, posted by Andrés C. Rodríguez (Argentina) on 7 June 2002, 6:11 p.m.*

The method I used was actually a little simpler (in theory, anyway) than that mentioned by Mr. Brogger. It involved copying the two status registers ("a" & "b" as I recall), which contain all six return pointers and the program counter, to main memory, starting at an address that the user specified in an initialization routine. Two other registers were also used to keep track of the number of return frames (set of six return pointers) that had been used and the current reference count within each frame (so that the routine knew when it was time to copy the current frame and start a new one), and a third register to keep track of the starting point. I tried using one of the alpha registers for the last, but every once in a while something would happen and a program would munge the value. Lossage would then ensue.

It worked -- to a point. One of the problems I ran into involved the program counter, and another problem was (as is often the case with reference-counting systems) with calls to routines that didn't play along. The details are a little fuzzy in my head since I haven't really thought about this for 15 years or so.

(Wow. High weirdness in the forums.)

Final Remarks

Posted by ***Ex-PPC member*** *on* ***8 June 2002, 11:35 p.m.****, in response to Short & Sweet Math Challenges for HP fans #2, posted by Ex-PPC member on 1 June 2002, 9:27 p.m.*

Thanks to all those people that posted and/or were interested in this thread. This time there were fewer solutions given and no HP-calc code was produced, but I hope that at least you enjoyed the challenged. Now, just some final notes:

- Mr. Ioannidis was able to find the maximum value for the 3x3 determinant, 412. This is indeed correct.
- The minimum *non-zero* value es 1. One such arrangement is:

```
|  1  2  3  |
|  7  5  8  |  =  1
|  4  6  9  |
```

- The minimum *non-zero* value for the 4x4 determinant is also 1. One solution is:

```
|  1  2  3  4  |
|  5  6  7  9  |
|  8 10 11 12  |  =  1
| 16 15 13 14  |
```

  A more economic notation for the above solution is:

```
det(1,2,3,4,5,6,7,9,8,10,11,12,16,15,13,14) = 1
```

- As for the additional, miscellaneous questions, here are two solutions:

```
det(1,2,3,4,5,6,16,8,12,15,10,9,14,7,11,13) = 4000
det(1,2,3,4,5,8,14,6,11,16,9,10,15,7,12,13) = 4444
```

- Finally, the solution to the maximum value for the 4x4 determinant, I'll leave that for you. For instance, the following arrangement gives a suitably high value, but ... is it the maximum possible value ? That's for you to find out:

```
det(16,9,7,4,8,1,13,11,3,12,15,5,6,10,2,14) = 39030
```

- Also, many of you were interested in recursion. Actually, it has little to do with this challenge, as the most efficient procedure does not entail using recursion at all, but it is an interesting topic in itself.

  As was stated, some HP calculators do support recursive procedures natively, such as the 71B BASIC, which supports recursive subprograms and user defined functions, and also the RPL models, which by the very nature of their programming model being based on a stack do also permit recursion very easily.

  What to do when a programming language does not support recursion ? Simple, you fake it. For instance, in the 41C normal recursion would be limited to 6 levels deep if using XEQ. So the answer lies in not using XEQ (or GOSUB, GSB, etc. in other HP models), which uses the internal, limited return stack, but simply to use that much-dreaded instruction, GOTO (GTO, etc), together with a fake "return stack", created and maintained by ourselves, typically in some structure like an array, or some suitable registers put aside for this.

Does this sound difficult, or does it require complex, lengthy programming to achieve it ? Far from it, it's extremely simple, almost trivial. It has been mentioned here that the "Towers of Hanoi" program would be a suitable test case, and it is. As an example, suppose we were to implement this using a version of BASIC that does not support recursion. As the only HP calculator that includes BASIC is the HP-71B and it does support recursion, let's use some other version, say the most simple, limited BASIC of them all, the one used in a famous contemporary of the HP-41C, the original Tandy Radio Shack TRS-80 Pocket Computer (exact clone of the Sharp PC-1211 Pocket Computer). This is a perfect machine for the experiment, as its BASIC is as simple as it gets, it has only 1.4 Kb of memory, and very few commands, not even multiple arrays or two-dimensional arrays.

Well, in such limited machine/language you can use the simulated-return technique to program the entire "Towers of Hanoi", simulated recursion and all, in just *six* (6) lines of BASIC. That qualifies as absolutely trivial. You can try the technique in your favorite HP calc as well, see how many lines/steps/bytes does it take you to program ToH simulating recursion. You can test your program running this example, which on the TRS-80 PC-1 goes like this (N is the number of disks, limited only by available memory, i.e: more than 50 disks possible)

```
RUN
N=3
FROM 1 TO 3
FROM 1 TO 2
FORM 3 TO 2
FROM 1 TO 3
FROM 2 TO 1
FROM 2 TO 3
FROM 1 TO 3
DONE
```

Re: Final Remarks

*Posted by **Chris Randle (Lincoln - UK)** on **9 June 2002, 6:00 a.m.**, in response to Final Remarks, posted by Ex-PPC member on 8 June 2002, 11:35 p.m.*

I'm still thinking about this puzzle! My brain is slower than most!!

I hope to have some code soon, so could you give me another 24 hours to come up with something (like code that can find the answers) before posting any solution? I mean algorithmic solution - not numeric (which has already been found).

Re: Final Remarks

*Posted by **Ex-PPC member** on **9 June 2002, 6:52 p.m.**, in response to Re: Final Remarks, posted by Chris Randle (Lincoln - UK) on 9 June 2002, 6:00 a.m.*

Chris Randle wrote:

"I'm still thinking about this puzzle! [...] could you give me another 24 hours to come up with something (like code that can find the answers) before posting any solution?"

Thanks for your interest, and don't worry for any deadline to develop your solution, as I rarely post my 'code' solutions, if ever.

Normally I'll post a "Final Remarks" message, which may contain some numeric solutions and/or interesting related notes, hints and references, and that's it.

Short & Sweet Math Challenges for HP fans #3

Posted by ***Ex-PPC member*** *on* ***9 June 2002, 12:43 a.m.***

Last week's challenge involved an empirical search for extreme values of determinants, which required to either own an HP calc with built-in matrix operations, or else to remember just how a 3x3 or 4x4 determinant could be calculated, which made it necessary for some of you to reach for that dusty math book on the shelf.

This time we have a new S&S Math Challenge requiring an empirical search, but the underlying concepts are much simpler, no "Matrix Calculus for Dummies" needed. I'll introduce the challenge with an example of what we are going to find out with the help of our precious, museum-quality HP calcs [if you don't use them, they'll dissecate and die, you know]:

Have a look at the number 153. Does it have any interesting peculiarity ? Well, it does have several, the most remarkable one being that:

```
                153 = 1^3 + 5^3 + 3^3
```

i.e.: 153 is a 3-digit number equal to the sum of the 3rd powers of its own digits. Likewise, have a look at 1634, and you'll see that:

```
                1634 = 1^4 + 6^4 + 3^4 + 4^4
```

this is, 1634 is a 4-digit number equal to the sum of the 4th powers of its digits. Let's generalize this to N-digit numbers and so we have the following challenge:

1. Pick up your favorite HP calculator. Almost any will do for the task at hand, as it does not involve anything but arithmetic functions on integer numbers, and negligible memory requirements. Do *not* use any computer for this, the challenge is tailored for the speed and capabilities of HP calculators and using a computer just misses the point by a light-year or more.

2. Find *all* numbers of N digits that are equal to the sum of the N-th powers of their digits. You must find the numbers themselves, as well as just how many there are for each N. The digit 0 *won't* be accepted as first digit, so for instance 153 may be a solution for N=3 but 032 wouldn't even be tested.

3. You should determine all solutions for N=1,2,..., 10. The lower values of N, say up to N=4 are easy and feasible for any HP calculator, from the HP-25 upwards, even using a straightforward programming approach.

4. However, for N=5 to N=10, you'll need to refine your approach significantly if you intend to get results in a reasonable amount of time.

5. For test purposes, this is what you should get:

6.	N	# Solutions	Solutions	Comments
7.	---			
8.	1	9	1,2,3,4,5,6,7,8,9	0 is not acceptable
9.	2	0	none	00 and 01 not acceptable
10.	3	4	153, 370, 371, 407	easy
11.	4	3	1634, ?, ?	easy
12.	5	3	?,?,?	still easy
13.	6	1	?	less easy
14.	7	4	?,?,?,?	medium
15.	8	3	?,?,?	hard
16.	9	4	?,?,?,?	very hard
17.	10	?	?	very, very hard

That's all. May I repeat once more: do *not* use your computer, use your HP calculator. It is much more difficult, but hey, that's where the challenge is. At least see if you can find the ONLY, unique solution for N=6 !

Of course, the real 48/49 nuts among you should try up to N=10, using Saturn Assembler if necessary.

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **Thibaut.be** on **10 June 2002, 5:56 a.m.**, in response to Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 9 June 2002, 12:43 a.m.*

Hello,

I programmed a routine on my 41C. To be very honest I first tested it on my computer . I let it run all night and these are the results I found :

n = 4 : 1.634,8.208,9.474 n = 5 : 92.727,93.084 n = 6 = is till running.

Now, I guess there is something to deduct about the series. I'll be thinking about it, but it seems that if my 41C can find in looping the only n=6 solution, it is practically impossible to find the n>6 solutions with a 41...

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **Thibaut.be** on **10 June 2002, 10:07 a.m.**, in response to Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 9 June 2002, 12:43 a.m.*

For n=6 : what about 548834 ?

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **Andrés C. Rodríguez (Argentina)** on **10 June 2002, 6:02 p.m.**, in response to Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 9 June 2002, 12:43 a.m.*

This time I'm finding a little time to work in this stimulating challenge. Two questions:

1) Would you expect us to post here our code to solve the challenge, or just the numerical answers?

2) May the digits repeat in the number? (for instance, a number like 1232 would have been a valid answer if 1+16+81+16 equals 1232 (which of course does not!))

I already have a program running in my HP42S (using just 7 registers and less than 50 steps, even with a little error checking and sort of a "user interface")which works very well for numbers between 1 and 4 digits; while it also works for N=5 and greater, execution time will be more than excessive, so I am exploring a different approach...

As a reference, it took some 25 minutes to find all solutions for N=3

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **Ex-PPC member** on **10 June 2002, 7:13 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #3, posted by Andrés C. Rodríguez (Argentina) on 10 June 2002, 6:02 p.m.*

Some quick answers to the questions raised:

Mr. Thibaut.be wrote:

"I programmed a routine on my 41C [...]I let it run all night and these are the results I found :

```
n = 4 : 1.634, 8.208, 9.474
n = 5 : 92.727, 93.084
n = 6 = [ ...] what about 548834 ? "
```

Your results are correct, except for the fact that it seems you missed the 3rd solution for N=5. An unwanted omission, or else a bug in your code ? As for the result for N=6, it would be interesting to know how long did your 41C took to find it. Will you try N=7 on the 41C or other faster machine ? (I would suggest a 42S, as the code should run unchanged, and it's 10x to 14x times faster. An HP32S or 32SII would be a good choice, too, as they are faster than the 42S, albeit less compatible with the 41C).

Mr. Andres C. Rodrigues wrote:

"Would you expect us to post here our code to solve the challenge, or just the numerical answers?"

As you like. I guess that posting code is Ok as long as it's not too large and it does use some interesting technique that other users may find worthwhile for learning purposes, or for discussing alternatives. Another possibility is to offer guidelines or hints that others may find useful to develop their own code. If code is too large or too straightforward, I guess that posting just the solutions would be preferable.

"May the digits repeat in the number?"

Yes, they may, and often do.

"I already have a program running in my HP42S [...] for N=5 and greater, execution time will be more than excessive, so I am exploring a different approach [...] As a reference, it took some 25 minutes to find all solutions for N=3"

I thank you for your interest in this challenge, Mr. Rodrigues, and please don't take me wrong, but 25 minutes for N=3 in a 42S is *far* too slow. You should definitely need to explore that different approach you mention :-) That is, unless it's a typo and you really meant 25 seconds.

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **Andrés C. Rodríguez (Argentina)** on **10 June 2002, 8:09 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 10 June 2002, 7:13 p.m.*

Ok, thank you... I am even more engaged by the challenge. It was 25 minutes, and rather brute-force based programming, I admit...

While the "other" approach is promising, I still have to find a way to generate certain "patterns" of numbers, something very easy to do by hand, but not that easy so far to convert in HP code. It also may happen to be more elegant but slower than the first method!

Re: Short & Sweet Math Challenges for HP fans #3

*Posted by **thibaut.be** on **11 June 2002, 8:35 a.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 10 June 2002, 7:13 p.m.*

While a loop instruction does not seem to be efficient for values aboce 10^5, I was wondering about expressing this problem in a mathematical way.

Actually I also recalled my linear programmation maths course and the SIMPLEXE algorithm. Have you heard about it ? It is used in economics to calculate maximums or minimums of a linear function under constraints. This is exactly what this is about excepted that the function is not linear.

for instance, for N=5, you have to solve the equation

10.000a + 1.000b + 100c + 10d + e = a^5 + b^5 + c^5 + d^5 + e^5

WHERE

1<=a<=9 0<=b<=9 0<=c<=9 0<=d<=9 0<=e<=9

and a;b;c;d;e are integers

Does this track is the correct one or is it time loosing searching this way ?

Clarification

*Posted by **Andrés C. Rodríguez (Argentina)** on **11 June 2002, 1:26 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #3, posted by Andrés C. Rodríguez (Argentina) on 10 June 2002, 6:02 p.m.*

It takes about seven minutes to find 153, 370, 371 and 407. It needs more time to verify there are no more solutions. Still improvable...

What do you think about that !!!!!!

Posted by ***Frédéric ALBERT (from France)*** *on* ***11 June 2002, 6:47 a.m.****, in response to Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 9 June 2002, 12:43 a.m.*

1 : 1 2 3 4 5 6 7 8 9

2 :

3 : 153 370 371 407

4 : 1634 8208 9474

5 : 54748 92727 93084

6 : 548834

7 : 1741725 4210818 9800817 9926315

8 : 24678050 24678051 88593477

9 : 146511208 472335975 534494836 912985153

I have some difficulties for 10 but i am still working on .... I will post the code for hp49g when found !!!! If found ;-(

Fred

Re: What do you think about that !!!!!!

Posted by ***thibaut.be*** *on* ***11 June 2002, 8:24 a.m.****, in response to What do you think about that !!!!!!, posted by Frédéric ALBERT (from France) on 11 June 2002, 6:47 a.m.*

Félicitations... et quel genre de routine

(Congratulations... and what kind of routine did you use ?)

Re: What do you think about that !!!!!!

Posted by ***Jordi Hidalgo*** *on* ***11 June 2002, 12:16 p.m.****, in response to Re: What do you think about that !!!!!!, posted by thibaut.be on 11 June 2002, 8:24 a.m.*

Maybe he cheated ... The complete solution
Pickover's book "Wonders of Numbers" is really wonderful.

Regards,
Bye.

Jordi Hidalgo
HPCC member #1046

Re: What do you think about that !!!!!!

*Posted by **thibaut.be** on **13 June 2002, 6:22 a.m.**, in response to Re: What do you think about that !!!!!!, posted by Jordi Hidalgo on 11 June 2002, 12:16 p.m.*

Well, no reaction from Mr ex-PPC Member ?

I really wonder if calculator program solutions rather than a loop exist. Though the link shown by Jori explains that what we were looking for are narcissitic numbers, no method to compute them was given.

For n=3 my CV took 14 minutes. So for n=10 it should take 14 minutes * 10^7, or a bit more than 266 years. So, Mr Ex-PPC Member, what is the solution ?

Re: What do you think about that !!!!!!

*Posted by **Andrés C. Rodríguez (Argentina)** on **14 June 2002, 7:27 a.m.**, in response to Re: What do you think about that !!!!!!, posted by thibaut.be on 13 June 2002, 6:22 a.m.*

With a looping program with some "improvements", it took my 42S less than 4 minutes to find the 4 solutions for N=3, but it took almost 12 minutes to be sure there are no other solutions. While I have some ideas to try (not sure about them), I have had not enough free time during the week, I would like to have an extra couple of days to see what may happen before the answer is given by Mr. Ex-PPC.

Re: What do you think about that !!!!!!

*Posted by **Fernando del Rey** on **14 June 2002, 5:27 p.m.**, in response to Re: What do you think about that !!!!!!, posted by Andrés C. Rodríguez (Argentina) on 14 June 2002, 7:27 a.m.*

I've tried on my HP-71B with a straightforward looping routine, also with some pretty trivial "improvements", and it takes about 320 minutes to find the solution for N=6.

But of course, there's no merit in doing it in BASIC and by "brute force". An this approach will not do the job for N>6 in a reasonable time. There has to be a much more clever way to solve the problem.

By the way, Mr. Hidalgo (a few posts earlier in this thread), please don't spoil the fun for the rest of us by giving the source of the solution away so quickly.

S&SMC #3: Final Remarks [LONG!]

*Posted by **Ex-PPC member** on **14 June 2002, 11:17 p.m.**, in response to Short & Sweet Math Challenges for HP fans #3, posted by Ex-PPC member on 9 June 2002, 12:43 a.m.*

Thanks to all of you who were interested in my S&SMC #3, the challenge is over and here are some final notes and remarks:

As has been mentioned in a post, the numbers who are equal to some mathematical function of their digits are generally referred to as *"Narcissistic Numbers"*. In Challenge #3, we were looking for numbers of N digits that are equal to the sum of the N-th powers of their digits. In the mathematical literature, these are known as *"PluPerfect Digital Invariants"*, i.e: PPDI. They can be defined for bases other than 10, but in what follows, base 10 will always be assumed, and we'll call Nth-order PPDI to a PPDI having exactly N digits. A lot is known about them, for example:

1. There is only a *finite* number of PPDIs, exactly 88. It is very easy to demonstrate that their number is finite, but finding that there are exactly 88 of them does require a lot of ingenuity and computer muscle.

2. There are *no* PPDIs of order N when N=2,12,13,15,18,22,26,28,30,36 nor for N>39, so 39 is the largest possible order.

3. There is *one and only one* PPDI when N=6,10,14,20,32,34,37,38

4. The *largest*, 88th PPDI of order N=39 is the 39-digit number:

   115132219018763992565095597973971522401

   which is equal to the sum of its digits raised to the 39th power.

5. The largest *prime* PPDi is the 23-digit numer:

   35452590104031691935943

   which is equal to the sum of its digits raised to the 23rd power.

The solutions for S&SM Challenge #3 are all PPDI of orders N from 1 to 10, i.e:

```
Order N             Solutions
-----------------------------------------------------
   1       1, 2, 3, 4, 5, 6, 7, 8, 9
   2       none
   3       153, 370, 371, 407
   4       1634, 8208, 9474
   5       54748, 92727, 93084
   6       548834
   7       1741725, 4210818, 9800817, 9926315
   8       24678050, 24678051, 88593477
   9       146511208, 472335975, 534494836, 912985153
  10       4679307774
```

Can our beloved HP calculators find these solutions ? Yes, of course. As I said, almost any model from the HP-25 onwards can find all solutions up to N=4 with relative ease, even using a straightforward brute-force approach. Finding solutions for orders N=5 to N=10 in a reasonable amount of time does require a combination of faster models, such as the HP-71B, HP-32S/SII, HP-42S or HP-48/49, as well as much more refined, optimized programming, and even a completely different approach.

Let's see an example of how can you proceed in order to achieve the goal. In this example, we'll compute all 3 solutions for order N=4, using an HP-71B, as its BASIC language helps to make the programming easier to understand. We'll start with a vey simple brute-force approach, then we will refine it stepwise, creating better and faster versions which will allow us to reach higher orders. Then, once we reach the limits of our direct approach, we will hint at a newer, much faster approach which will deliver what we want. Then, a direct conversion of the HP-71B BASIC code to the HP-41C's native RPN will be shown.

Caveat emptor: You should bear in mind that, in all cases, the programs shown have been produced strictly for demonstration purposes and are not meant to be the ultimate programming solution for this challenge, or even highly optimized, so I don't claim or intend them to be state-of-the-art programming at all.

Program P1-71B:

As we are looking for all 4-digit numbers equal to the sum of the 4th powers of its digits, this means we are looking for numbers which are solutions of this Diophantine equation:

[ABCD] = 1000*A+100*B+10*C+D = A^4+B^4+C^4+D^4

where A goes from 1 to 9, and B,C,D all go from 0 to 9. A very simple program, made essentially of four nested FOR-NEXT loops is quickly produced:

```
10 DESTROY ALL @ DIM A,B,C,D @ STD @ DELAY 0,0
20 FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF A^4+B^4+C^4+D^4=1000*A+100*B+10*C+D THEN DISP A;B;C;D
40 NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

When run, this program displays all three solutions [1634, 8208, 9474], then terminates displaying "DONE" after verifying there are no more.

Running time: T = 2050 sec.

Program P2-71B:

One of the easiest ways to reduce the running time of a program is to simplify the computations within the innermost loop. In this case, program P1 is continually evaluating the expression:

1000*A+100*B+10*C+D

inside the innermost loop. But a bit of thinking or a quick hand simulation will convince you that this is just the value of the number itself, which is being incremented by one in each iterarion. So we can simply initialize this value to 1000 (A=1, B=0, C=0, D=0), and then increment it after each iteration, thus saving a lot of computation:

```
10 DESTROY ALL @ DIM A,B,C,D,N @ STD @ DELAY 0,0
20 N=1000 @ FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF A^4+B^4+C^4+D^4=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 1854 sec. (1.11 times faster than P1)

Program P3-71B:

A further refinement comes when we realize that we are continually computing the 4th powers of the digits of N inside the innermost loop. But there are only 10 distinct values, namely 0^4, 1^4, ..., 9^4, so we can save a lot of time pre-calculating them outside of all loops and storing them on a suitably dimensioned array, then simply recalling the values of the powers when needed, instead of computing them anew each time. As it is much faster to recall an array element than to raise a number to a power, great time savings are to be expected:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A
20 N=1000 @ FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF P(A)+P(B)+P(C)+P(D)=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 460 sec. (4.46 times faster than P1)

Program P4-71B:

Is that all ? Can't we still do more optimization ? Yes. Look at the IF at line 30. We are constantly recalling and adding up the powers of the digits to compare them against the value of N. But it's clear that inside the FOR D=... loop, the value of P(A)+P(B)+P(C) is invariant, as it does not depend on the loop variable, D. Same applies to P(A)+P(B) inside the FOR C loop, and P(A) inside the FOR B loop. Keeping this is mind, we avoid recomputing sums and invariant values repeatedly, but we'll simply compute them once outside of the respective loops, and store them in intermediate variables, like this:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,S,T,U,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A @ N=1000
20 FOR A=1 TO 9 @ S=P(A) @ FOR B=0 TO 9 @ T=S+P(B)
30 FOR C=0 TO 9 @ U=T+P(C) @ FOR D=0 TO 9 @ IF U+P(D)=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 320 sec. (6.41 times faster than P1)

Program P5-71B:

There's still one trick out of our sleeve. Just notice what happens when, for instance, we reach the inner FOR C and FOR D loops when A=9 and B=8. At this point, the sum is already 9^4+8^4 = 10657, which exceeds the maximum possible value for N, namely N=9999. So, further adding the 4th powers of C and D is no use, since the resulting sum can never equal N. This means we can skip altogether the full FOR C and FOR D inner loops in this case, so saving 100 full iterations. Applying this simple idea gives us the program:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,S,T,U,V,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A @ N=1000
20 FOR A=1 TO 9 @ S=P(A)
25 FOR B=0 TO 9 @ T=S+P(B) @ IF T>N THEN N=N+(10-B)*100 @ GOTO 80
30 FOR C=O TO 9 @ U=T+P(C) @ IF U>N THEN N=N+(10-C)*10 @ GOTO 70
40 FOR D=0 TO 9 @ V=U+P(D) @ IF V>N THEN N=N+10-D @ GOTO 60 ELSE IF V=N THEN
PRINT N
50 N=N+1 @ NEXT D
60 NEXT C
70 NEXT B
80 NEXT A @ DISP "DONE"
```

Running time: T = 242 sec. (8.47 times faster than P1)

[By the way, just in case you are wondering, declaring the variables with INTEGER precision instead of floating-point DIM is actually *slower*! HP-71B BASIC always works with floating-point variables internally, so declaring them INTEGER just adds a lot of conversions to floating-point precision and back to integer]

So you see, applying just a few, simple common sense ideas to our initial try has resulted in a program that, while still being very simple and not much larger, nevertheless runs nearly an order of magnitude faster. Which is better, the savings are even greater for larger orders of N, so that this approach allows us to compute all solutions (and to certify there are no more) in these running times:

Order N	P1	P2	P3	P4	P5
3	02:32	02:21	00:40	00:32	00:25
4	34:10	30:54	07:40	05:20	04:02
5	-	-	-	-	37:20
6	-	-	-	-	5:07:13

Remember, these times are measured from start to until the program stops by itself, not merely until the last solution is displayed. Times for an HP-32S/SII, HP-42S or HP-48/49 would be even better, as their CPUs are much faster than the old 640 Khz Saturn CPU of the 71B.

However, for N=7 to N=10 our current approach does not work in reasonable times, and we find we have two possible options:

- a) stick to our current approach, but change to a faster language, using for instance FORTH instead or BASIC, or better still, using Saturn Assembler language (using the 71B Forth/Assembler ROM, for instance).

  Converting our program P5 to Saturn assembler is not difficult at all, and we can take advantage of fast custom-made all-integer routines, much faster than the floating-point ones we're using in BASIC. The resulting *binary subprogram* (not BASIC keyword) is faster than our BASIC version by two full orders of magnitude, and so allows us to go up to N=7 in under 30 min., N=8 in some 4 hours, and N=9 in roughly a day and a half continuous running time. The final summit, N=10, would take more than 2 weeks, so this approach's usefulness ends at N=9.

- b) change our current approach for a completely different one. To that effect, we notice that (for our example, N=4) we are exhaustively searching all arrangements from 1000 to 9999, using 4 nested loops, so a total of 9000 cases are tested in all. Our trick in P5 avoids some of these, but only to the point were we have an exponential increase of roughly 8.5x per additional digit instead of 10x. Useful but insufficient.

  However, *we are testing far more cases than we actually need !*. For instance, for N=6754, we are computing the sum 6^4+7^4+5^4+4^4 and comparing it versus 6754. But the sum itself is invariant whatever the order of the digits may be: we have Sum4(6754) = Sum4(6745) = Sum4(4567) = ...

So, the magic word is *lists*: you just need to test all *different* lists, *order doesn't matter !!* This tremendously reduces the number of cases we need to test, to the point where, with clever list-generating programming, we can succeed even for N=10 ! And, with the help of a fast PC, we can even find all 88 PPDIs, up to order 39, in a reasonable amount of time.

Of course, I won't give here the solution using this lists approach, that's for you to take over. But for the sake of all of you who tried this challenge on your HP-41Cs or your HP-42S, I'm including a straight conversion of P4 to HP-41C/CV/CX's RPN, for the case N=3:

Program P4-41C:

```
01  LBL "N3"       16  DSE 16          31  RCL 14
02    9            17  LBL 01          32    +
03  STO 00         18  RCL IND 10      33  RCL 15
04  LBL 00         19  STO 13          34   X=Y?
05  RCL 00         20  RCL 16          35  VIEW X
06    3            21  STO 11          36  ISG 15
07   Y^X           22  LBL 02          37  LBL 04
08  STO IND 00     23  RCL IND 11      38  ISG 12
09  DSE 00         24  RCL 13          39  GTO 03
10  GTO 00         25    +             40  ISG 11
11   100           26  STO 14          41  GTO 02
12  STO 15         27  RCL 16          42  ISG 10
13  1.009          28  STO 12          43  GTO 01
14  STO 10         29  LBL 03          44  "DONE"
15  STO 16         30  RCL IND 12      45  PROMPT
```

This 45-line program will compute all four solutions for N=3, in these times:

```
  Time to find and display 1st solution (153) ....         24 sec.
                           2nd solution (370) ....  1 min 37 sec.
                           3rd solution (371) ....  1 min 37 sec.
                           4th solution (407) ....  1 min 49 sec.
                           "DONE" ................  5 min  6 sec.
```

This program will also run unchanged on an HP-42S, only much faster.

That's all. Next S&SMC #4 soon, stay tuned !

Re: S&SMC #3: Final Remarks

*Posted by **Andrés C. Rodríguez (Argentina)** on **15 June 2002, 2:49 p.m.**, in response to S&SMC #3: Final Remarks [LONG!], posted by Ex-PPC member on 14 June 2002, 11:17 p.m.*

Thank you, Mr. Ex-PPC Member for a nice challenge, I hope next time I will be able to do it better. My best program incorporated some of the techinques you suggested, but I haven't had enough time during the week to fully polish it. For N=3, I needed about 4 minutes to reach 407, and 12 minutes to exhaust all possibilities. Your example is about four times as fast, taking 3 minutes 12 seconds in my HP42S. By the way, the use of registers and steps count was similar in my case.

I was trying a shortcut by means of aborting the inner loop as soon as the sum of the powers exceeded the number under test, since the function is monotonic inside the inner loop. I also stored the differences between the powers in an array, instead of the powers themselves; so a simple RCL + IND 10 updated the X register with the next value to try. At the loop exit, I substracted 9**n from the sum-of-powers last value, and went to the next loop.

A binary search inside the inner loop, halving the range on each iteration may reduce the tests from 10 to 4, but I would lose my job (and-or my family) if I spend that much time and dedication from Monday to Friday. If I find reasonable way to do that in an HP42S, I will let you know.

Thank you again, and keep the challenges coming, please!

Re: S&SMC #3: Final Remarks [LONG!]

*Posted by **Andrés C. Rodríguez (Argentina)** on **16 June 2002, 8:32 a.m.**, in response to S&SMC #3: Final Remarks [LONG!], posted by Ex-PPC member on 14 June 2002, 11:17 p.m.*

While I know the challenge is over, I finished my HP 42S program, using recursive loops. It asks for M (the order of the problem), between 3 and 10, and then proceeds to find all solutions and to exhaust the possibilities.

For M=3, it took 1 minute 46 seconds to finish.

For M=4, it took 17 minutes.

For M=5, it took 3 hours.

The program uses some 420 bytes of program memory, and runs with 24 registers.

I also find that the generality of a program (opposed to write it just to solve a particular value of M) conspires against its speed.

As I am not familiar enough with the Forum formatting techniques for such a long listing (6 pages), I offer to send a RTF text file with Introduction, Code and Comments to anyone interested, who sends me his-hers email address. Or, after some extra time, I may be able to publish it here.

My program (very long)

*Posted by **Andrés C. Rodríguez (Argentina)** on **16 June 2002, 8:46 a.m.**, in response to Re: S&SMC #3: Final Remarks [LONG!], posted by Andrés C. Rodríguez (Argentina) on 16 June 2002, 8:32 a.m.*

Well, this is my first attempt at formatting, not 100% readable,

Author: Andrés C. Rodríguez Date: June 16, 2002 Calculator: HP 42S Size=25

Principle of operation:

Accepts “M”, which is the number of digits searched for. There are recursive loops for each digits position.

The Units loop contains the test statements, to check if the Number Under test (NUT) equals the Sum of Powers (SOP) of its digits.

Since the Units loop is monotonic, an extra test is used to abort the loop when there are no more solutions in the decade under test.

To obtain greater speed, this program uses an array (R0-R9), which contains the differences between the powers to be added. The last value of such array is negative, allowing for an easy loop-back.

A Diagnostics routine is provided, which slows the program but helps in debugging or understanding, showing the progression of the values under test.

This program fully uses the 8-level pending subroutine return addresses of the HP42S, hence it needs to be adapted for other calculators like the HP 41 family. The Units routine is called via GTO and has fixed return addresses because of this limitation.

In some cases, a value for NUT already incremented is decremented upon the loop exit, since the calling, outer loop would increment it again (like the carry in addition) causing an erroneous condition.

Based upon the “M” value, the program starts at the proper looping routine. The outermost routine loops between 1 and 9, all others loop between 0 and 9.

While it uses recursive loops, a difference array, a shortcut for Units loop abort and a lot of inline code for speed, I still consider it a “Brute Force” approach.

Special symbols:

The | symbol means “line feed” for nicer Alpha displays The words “Roll Down” are used for the so-called function The symbol is the Alpha Append character

```
LBL “PPC3R”            ; Initialize
CLRG
CLST
FIX 0
“Enter M (3 10)|”      ; Get M (| = line feed)
PROMPT
CLLCD
IP                             ; M must be an integer between 3 and 10
3
x>y?
```

```
GTO 26
Roll Down                     ; Just for clarity, I write it this way...
10
x<y?
GTO 26
Roll Down
STO 20
0.009
STO 23                ; Constant used for loop initialization
9
STO 22                ; R22 will be reused later on
LBL 22                ; Initialize array of powers differences
RCL 22
1
-
RCL 22
RCL 20
y^x
STO - IND 22
STO + IND Y
DSE 22
GTO 22                ; Differences are in R0 – R9

; Now, in R(i) we have  [(i+1)**m – i**m]
               ; For instance, for m=3, we should have:

; R0 = (1-0) = 1        R1 = (8-1) = 7
; R2 = (27-8) = 19      R3 = (64-27) = 37
; R4 = (125-64) = 61    R5 = (216-125) = 91
; R6 = (343-216) = 127  R7 = (512-343) = 169
                      ; R8 = (729-512) = 217  R9 = (0-729) = -729 (!)

10                            ; Continue initialization
RCL 20
1
-
y^x                           ; Calculate 10^(m-1) as initial value, to
STO 21                ; initialize R21 with Number-Under-Test (NUT)
1                             ; Use 1 to
STO 22                ; initialize R22 with Sum-of-Powers (SOP)

RCL 20                ; ***** S T A R T *****
9
+
1.009                         ; The outermost loop starts with 1
XEQ IND Y                     ; Entry point of nested routines based on M.
“DONE |”                      ; When back, prepare final message,
CLST                          ; be polite by clearing the stack,
BEEP                          ; and salute
PROMPT
GTO “PPC3”                    ; Start over
LBL 26                ; Improper M values
“Bad M |”
TONE 0
PROMPT                ; Warn the user
GTO “PPC3”                    ; Start over


LBL 19                        ; The following pages contain very
STO 19                        ; similar, nested routines, which
LBL 09                ; are used to cycle thru each digit
RCL 23                ; position.
XEQ 18                ; See comments on LBL 12, the one which
```

```
RCL IND(19)            ; takes care of the hundreds digit.
STO + 22                       ; LBL 19 to LBL 12 are essentially the
1       ; same.
2       ; The calls are sort of recursive, from
STO + 21                       ; the most significant digits to the
ISG 19                 ; lesser ones.
GTO 09                 ; The internal labels and addresses are
STO –21                ; shifted from one routine to the next.
RTN

LBL 18
STO 18
LBL 08
RCL 23
XEQ 17
RCL IND(18)
STO + 22
        1
STO + 21
ISG 18
GTO 08
STO – 21
RTN

LBL 17
STO 17
LBL 07
RCL 23
XEQ 16
RCL IND(17)
STO + 22
        1
STO + 21
ISG 17
GTO 07
STO –21
RTN



LBL 16
STO 16
LBL 06
RCL 23
XEQ 15
RCL IND(16)
STO + 22
        1
STO + 21
ISG 16
GTO 06
STO – 21
RTN

LBL 15
STO 15
LBL 05
RCL 23
XEQ 14
RCL IND(15)
STO + 22
        1
STO + 21
```

```
ISG 15
GTO 05
STO –21
RTN

LBL 14
STO 14
LBL 04
RCL 23
XEQ 13
RCL IND(14)
STO + 22
       1
STO + 21
ISG 14
GTO 04
STO – 21
RTN


LBL 13
STO 13
LBL 03
RCL 23
XEQ 12
RCL IND(13)
STO + 22
       1
STO + 21
ISG 13
GTO 03
STO –21
RTN

LBL 12                 ; Hundreds loop
STO 12                 ; Get initial value from calling routine
LBL 02         ; Inner hundreds loop starts here
RCL 23         ; Use 0.009 parameter
XEQ 11         ; to calls the decades routine
RCL IND(12)    ; Then, increase the hundreds digit
STO + 22               ; so SOP is properly incremented,
       1                       ; and also increment the NUT by one
STO + 21
ISG 12         ; Try again with the next hundreds digit
GTO 02
STO – 21               ; The thousands routine will increment it (!)
RTN                            ; After exhausting, return

LBL 11                 ; Decades loop (somehow different)
STO 11                 ; Get initial value from calling routine
LBL 01         ; Inner decades loop starts here
GTO 10         ; Branches to Units routine,
LBL 21         ; providing a return address (!)
RCL IND(11)    ; Then, increases the decades digit
STO + 22               ; so SOP is properly incremented,
       1                       ; and also increment the NUT by one
STO + 21
ISG 11         ; Try again with the next decades digit
GTO 01
STO –21        ; The hundreds routine will increment it (!)
RTN                    ; After exhausting, return
```

```
LBL 10                  ; Units routine
RCL 23                  ; Initialize index with 0.009
STO 10
RCL 21                  ; Recall NUT
RCL 22                  ; Recall SOP
        LBL 00          ; internal units loop
        x>y?                    ; If SOP > NUT, go to next decade
        GTO 25          ; (shortcut based on monotonicity)
        x=y?                    ; If SOP =     NUT, This is an answer!!
        VIEW X          ; Show answer without frills, and
RCL + IND (10) ; properly increase SOP in the stack         ,
        1                       ; and also increment the NUT by one
STO + Z         ; Just by now, we keep the next SOP in X,
Roll Down               ; and the next NUT in Y
ISG 10          ; Try again with the next units digit
GTO 00
STO 22          ; Only before returning, we update SOP
Roll Down               ; and NUT from the stack
STO 21          ; to the proper registers
GTO 21                  ; and go back to the decades routine
LBL 25          ; Remaining of this decade has no solutions,
9       ; increment NUT in R21 (!) by 9
STO + 21                ; Note: is not needed to increment SOP (!)
GTO 21                  ; Go back to the digits routine


LBL 27                  ; Diagnostics Routine, just for debugging.
CLA                             ; It would be called by XEQ 27, inserted just
ARCL Y                  ; after LBL 00. It slows the program.
“ “                             ; Appends a space
ARCL X
“ |”                            ; Appends a line feed
AVIEW                           ; Shows running values of NUT and SOP
RTN                             ; (A TONE 7 may replace VIEW X for tests)
```

Yes, I wrote it on a PC...

*Posted by **Andrés C. Rodríguez (Argentina)** on **16 June 2002, 8:52 a.m.**, in response to My program (very long), posted by Andrés C. Rodríguez (Argentina) on 16 June 2002, 8:46 a.m.*

... just as a text file. But I didn't run anything in the PC!

I just used the PC as a text editor, and then keyed the program to the HP42S. How nice a bidirectional interface would have been !!

Short & Sweet Math Challenges for HP fans #4 [LONG!]

Posted by ***Ex-PPC member*** *on* ***16 June 2002, 9:35 p.m.***

Welcome to Short & Sweet Math Challenges for HP fans #4. Not so "short" this time, but rather interesting, I hope ...

Foreword

Last week we were searching for integer numbers having some interesting property. This time, we depart from the integer realm and dive deep into the real numbers realm. As you know, many amazing things lurk in the depths, be it marine depths, interstellar space depths, whatever. Same thing for mathematics, once you stop scratching the surface and really go down, you may find lots of unexpected wonders waiting to be found. This week I'm challenging you to leave the comfortable built-in precission of your HP calculator, be it 10, 12, or 20 digits, and really go for multiprecission calculations, I'm talking 50 decimal digits ...

Sounds frightening ? Too difficult ? It shouldn't be. Back in the good old times of PPC, people were overcoming the built-in accuracy limitations of their HP-65s, HP-67s, and HP-41s, and writing wonderfully clever programs to perform computations to high-precission. Perhaps some of you will remember programs to compute the constant Pi to 1000+ digits and more on a 41C. Outside of PPC, you could find programs to compute multi-precission square roots and other functions in the HP User's Library. If people could do that with their 65s, 67s, and 41Cs back in the early eighties, there's no reason why you shouldn't try now, specially using the newer HP models, which boast much faster CPUs and much larger RAM.

So, I'm proposing you a challenge, were I will ask you to perform several computations to a prescribed high precission, then explain the amazing results you'll get. That will be the second half of the challenge. The first is you'll need to develop a multiprecission program, library or package to be able to perform said compuations on your HP.

You may beg, steal or borrow such a program, library or package, or you can write one yourself. For those of you wanting to try, I'll give some guidelines at the bottom of this posting. Be assured that this challenge is perfectly feasible, and even easy ! But you'll need a high performance HP calc. I think the barest minimum would be an HP-41CV, and you'll be much advised to use instead an HP-42S, HP-71B, or HP-48/49 for this task. FOr ease of presentation and explanation, all guidelines, examples and snippets of code will be given for an HP-71B's native BASIC language. They can be easily adapted/converted to RPN or RPL.

The Challenge (#4):

1. If you compute the square root of 308600 and 308699 to 41 decimal places, you'll get:

```
2.          sqrt(308600) = 555.51777649324598368631537686142297732063495
3.          sqrt(308699) = 555.60687540742330162311819242204090515640356
```

   which look as expected, long sequences of random-looking decimal digits, with no periodicity or pattern at all.

   But now compute the square root of 308642 to that same precission, 41 decimal places. Explain the amazing result. Can you find other numbers which behave like this ?

4. Compute 987654321/123456789 to 25 decimal places. Explain the result. In particular, is it just sheer coincidence that 729 (=9^3) which appears isolated at the 8th decimal position ?

5. Compute E^(PI*Sqrt(163)) to 12 decimal places (i.e: to 12 digits after the decimal point), where E = 2.718..., PI = 3.14159..., and Sqrt is the square root function. Explain the amazing result. Can you find any other values (instead of 163) behaving like this ?
6. This doesn't really require high precission, but it's funny: compute a real root for the following equation between 6 an 8 [Note: Log(x) is base-10 logarithm, not Ln(x)]. Give the root accurate to 12 decimal places:
7. `Log(x) - x - Sqrt(x-2) + Sqrt(19*x) = PI`

Optional: Guidelines for writing a multiprecission program/library/package for your HP calculator:

There are many ways to do it, but you can follow these simple guidelines. They aren't optimal, but will get the job done easily:

1. Chose a fixed-format representation for your multiprecission numbers, for instance
2. `(sign)50digits.50digits`
   i.e: 50 digits for the integer part, 50 digits for the decimal part, plus sign.
3. Store those digits internally in blocks of, say, 5 digits, in an array or consecutive storage registers. Such a number will occupy 20 elements, the position of the decimal point is implicitly assumed to be between blocks 10th and 11th.
4. Write a procedure to allow the user to input such a number, converting the user's input to the internal representation. Along the same line, write a procedure to convert a normal, 10- or 12-digit number as used in your HP calculator, to the internal representation for multiprecission.
5. Write a procedure to output a number in internal representation to a user-readable format. Similarly, write a procedure to convert a multiprecission value to the 10- or 12-digit format used natively by your calculator.
6. Write a procedure to change the sign of a multiprecission value
7. Write a procedure to add two multiprecission values and give the result as another multiprecission value. You just need to:
   - stablish a loop which will sum all corresponding blocks (taking into account the numbers' signs). As you are using 5-digit blocks, they will never overflow, as adding up two 5-digits numbers will only result in a 6-digit number at most.
   - stablish a second loop which will normalize all blocks to be 5-digits again, propagating carries from one block to the previous if its size was 6 digits.
8. Write a procedure to subtract two multiprecission values. You just need to call the procedure for changing the sign of the second one, then call the add procedure.
9. Write a procedure to multiply two multiprecission values. You just need to perform the multiplication as you would by hand, but using blocks of 5 digits at a time, and taking proper care of the carries, and of the final result. No intermediate overflows are possible, because multiplying two 5-digit numbers together will give a 10-digit result at most. Assuming your elements can hold 10 to 12 digits, there's no problem.
10. Write a procedure to divide one multiprecission number (A) by another (B). Here you can reduce the problem to multiplication, by using Newton's method to compute the

reciprocal of the second number (B): assuming x0 to be a good approximation to 1/B then a better approximation is x1, computed as

```
11.              x1 = x0*(2-B*x0)
```

which means we can compute reciprocals without division, just using multiplication. For instance, if x0=0.3 is an approximation to the reciprocal of B=3.7, then

```
x1 = 0.3*(2-3.7*0.3) = 0.267
x2 = 0.267*(2-3.7*0.267) = 0.2702307
x3 = 0.2702307*(2-3.7*0.2702307) = 0.27027026
```

are better approximations. The iterations converge quadratically, i.e: you get double the number of exact digits after each. As your initial approximation, you can use 1/B, which you can get to 10 or 12 digits by using the 1/X function of your HP calc ! Write a procedure to convert that value to the internal representation of multiprecission numbers, then compute 1/B using the iterative method above. As your initial 1/B is already accurate to 10 or 12 digits, just 2 or 3 iterations will give you 1/B accurate to in excess of 50 digits !

Once you've got 1/B to 50-digit precission, then A/B reduces to A*(1/B), which you can compute with just one extra multiprecission multiplication.

12. Write a procedure to compute the square root of a multiprecission value (A). Just use Newton's method once again:

```
13.              x1 = (x0 + A/x0) / 2
```

As your initial approximation, just use the value of SQRT(A) given by the square root built-in function. This will give you a value accurate to 10 digits, and then just 2 or 3 iterations of the procedure above will give you a square root accurate to in excess of 50 digits.

14. Include in your program the constant PI accurate to 50 decimal places. You can use this value:

```
15.             PI = 3.14159265358979323846264338327950288419716939937511
```

16. Finally, write a procedure to compute E^X where X is a multiprecission value. One possible way is to use its Taylor's Series Expansion, like this:

```
17.            E^X = 1 + X + X^2/2! + X^3/3! + X^4/4! + ... + X^n/n! + ...
```

where you should stop when the next term to add would be smaller than 1E-50. Of course, n! is the factorial function = 1*2*3*4*...*n

There are a number of tricks you could try to accelerate convergence. For instance, you could use the identity:

```
E^X = (E^(X/2))^2
```

i.e:, you can first divide X by 2, then compute E^(X/2) and square the result. As the argument X is smaller, convergence of the Taylor Series will be faster. Similarly, you could divide X by 8, then square the result three times in a row, etc.

Re: Short & Sweet Math Challenges for HP fans #4 [LONG!]

*Posted by **katie** on **17 June 2002, 12:29 a.m.**, in response to Short & Sweet Math Challenges for HP fans #4 [LONG!], posted by Ex-PPC member on 16 June 2002, 9:35 p.m.*

FYI,

I posted a program that I wrote (see the articles forum) some time ago to compute Pi to 99 digits on a 32SII. It uses many of the techniques that you mention for long precision arithmetic. Since it's only 99 digits the convergence time is pretty short (11 minutes), but the algorithm isn't particularly efficient just very compact (it needs to be given the severe memory limitation of the 32SII.)

-Katie

Begin of solution question #1

*Posted by **thibaut.be** on **18 June 2002, 10:38 a.m.**, in response to Short & Sweet Math Challenges for HP fans #4 [LONG!], posted by Ex-PPC member on 16 June 2002, 9:35 p.m.*

The inverse of 9, 90, 909009 and such numbers give interesting results : .11111, .011111, .000001100, ...

So a value of 555.5555578 is *almost* like 555.55555555, hence 5/9 * 1000 or 5000/9

If you square this value, you get 25 000 000 / 81 which is according to my CV 308 641.9753, or a very close value to 308 642.

Re: Begin of solution question #1

*Posted by **Fernando del Rey** on **26 June 2002, 5:59 p.m.**, in response to Begin of solution question #1, posted by thibaut.be on 18 June 2002, 10:38 a.m.*

The square root of 308642 is:

555.555577777773333333511111102222222227199999 ...

But I can't figure out why such a curious pattern of digits.

Any ideas from the clever members of this forum?

Re: Short & Sweet Math Challenges for HP fans #4 [LONG!]

*Posted by **Fernando del Rey** on **26 June 2002, 2:56 a.m.**, in response to Short & Sweet Math Challenges for HP fans #4 [LONG!], posted by Ex-PPC member on 16 June 2002, 9:35 p.m.*

I obtained the solution to Question #4 on my 42S and the result is:

x = 7.00000001061

Real close to 7!

The solver converged very quickly given the interval 6 to 8 to start the search.

Why this interesting result? I have no clue. I guess it's just a numerical curiosity, but perhaps Mr. Ex-PPC Member will give us an explanation.

Still no answers have been given to questions 2 and 3. Anyone still working on it?

Re: S&SMC #4: Final Remarks [LONG!]

Posted by ***Ex-PPC member*** *on* ***29 June 2002, 10:26 p.m.****, in response to Short & Sweet Math Challenges for HP fans #4 [LONG!], posted by Ex-PPC member on 16 June 2002, 9:35 p.m.*

Foreword

Well, it seems S&SMC #4 didn't catch the fancy of HP lovers this time, perhaps the subject matter wasn't interesting enough, or it was deemed too difficult or time-consuming for a casual approach. Certainly, most of us are always too busy to be able to commit scarce free time to everything we would like to, so I'm particularly grateful to **Mr. Thibaut** and **Mr. Del Ray** for their kind contributions.

That said, here are some answers to the questions raised in S&SMC #4:

Question 1.

> *If you compute the square root of 308600 and 308699 to 41 decimal places, you'll get:*
> ```
>             sqrt(308600) = 555.51777649324598368631537686142297732063495
>             sqrt(308699) = 555.60687540742330162311819242204090515640356
> ```
> *which look as expected, long sequences of random-looking decimal digits, with no periodicity or pattern at all.*
>
> *But now compute the square root of 308642 to that same precission, 41 decimal places. Explain the amazing result. Can you find other numbers which behave like this ?*

- **The raw facts:**

  Using your HP-41C and your multiprecision program, you first determine that, to 41 decimal places :

  ```
        sqrt(308642) = 555.55577777777733333333511111102222222227199999
  ```

  which indeed shows some *remarkable pattern*, unlike the other square roots above.

- **The explanation:**

  From the square root decimal expansion, it's clear that sqrt(308642) is very close to 555.5555... = 5000/9. So, with **d** being some small value, we have:

  ```
              308642 = (5000/9)^2 + d
  ```

  where d = 308642-(5000/9)^2 = 2/81, small as expected. Thus, we have:

  ```
              308642 = (5000/9)^2 + 2/81
  ```

  and factoring (5000/9)^2 out from the right side, we have:

  ```
              308642 = (5000/9)^2 * ( 1 + (2/81)/(5000/9)^2) = (5000/9)^2
  * (1 + 1/12500000)
  ```

  now, taking square roots in both sides:

  ```
        sqrt(308642) = sqrt[(5000/9)^2 * (1 + 1/12500000)] =
  (5000/9)*sqrt(1 + 1/12500000)
  ```

  Finally, our CAS-capable HP calc will give us the Taylor Series Expansion for sqrt(1+x):

  ```
        sqrt(1 + x) = 1 + x/2 - x^2/8 + x^3/16 + ...
  ```

  which for x = 1/12500000 yields:

  ```
        sqrt(1 + 1/12500000) = 1 + 1/25000000 - 1/1250000000000000 + ...
                             =
  1.0000000399999992000000319999984000000008959999...
  ```

and thus we have:

```
      sqrt(308642) = (5000/9) * sqrt(1 + 1/12500000)
                   = 555.555555555555... *
1.00000003999999920000003199999840000008959999...
```

and as we are multiplying together two heavily patterned numbers, it is only to be expected that their product will feature some pattern too, at least initially:

```
                   =  555.555577777777333333335111111022222227199999...
```

- **Finding more:**

  This will be left as an (easy) exercice. Just a hint: inspired by this particular example, write a short program for your HP to try and find periodic fractions n/9 such that (n/9)^2 comes as close to being an integer as possible. The square root of that integer will probably be an interestingly patterned number.

Question 2.

Compute 987654321/123456789 to 25 decimal places. Explain the result. In particular, is it just sheer coincidence that 729 (=9^3) which appears isolated at the 8th decimal position ?

- **The raw facts:**

  Using your HP42S and your multiprecision program, you find that:

```
     987654321/123456789 = 8.0000000729000006633900060368490...
```

  which is *very nearly* 8, except for the isolated 729, 66339, 6036849 ... Now, 729 = 9^3. This isn't a coincidence, as we will see.

- **Explanation:**

  Using our prime factor finder program or built-in function, we find that:

```
                                      729 = 9^3          = 9^3 * 91^0
                                    66339 = 9^3 * 91     = 9^3 * 91^1
                                  6036849 = 9^3 * 91^2   = 9^3 * 91^2
```

  so we can *conjecture* that:

```
         987654321/123456789 = 8 + 9^3*1E-10*SUM[N=0,N=INFINITE,(91*1E-
10)^N]
```

  where the sum goes from N=0 to N=infinite. This can be proved very easily using the well-known summation formula for geometric progressions, as this happens to be one.

Question 3.

*Compute E^(PI*Sqrt(163)) to 12 decimal places (i.e: to 12 digits after the decimal point), where E = 2.718..., PI = 3.14159..., and Sqrt is the square root function. Explain the amazing result. Can you find any other values (instead of 163) behaving like this ?*

- **The raw facts:**

  Using your HP-71B and your multiprecision program, you first find that, to 12 decimal places:

```
    E^(PI*Sqrt(163)) = 262537412640768743.999999999999...
```

  which comes *extremely close* to being an integer.

- **The explanation:**

  This time the explanation is more involved, has to do with the theory of modular functions, and can't be given here in full without making this already long post much longer. Interested people can have a look at this very interesting URL:

  http://membres.lycos.fr/bgourevitch/mathematiciens/approx/approx.html

  It's in French, but never mind, the numbers and formulas speak for themselves.

- **Finding more:**

  It's quite easy to write a program on your HP calc to test if Z=E^(PI*Sqrt(N)) is nearly an integer for various values of N. If you do write and run it, it should find these values:

  ```
       N =  25  -> Z =            6635623.9993+
         =  37  -> Z =          199148647.99997+
         =  43  -> Z =          884736743.9997+
         =  58  -> Z =        24591257751.9999998+
         =  67  -> Z =       147197952743.999998+
         =  74  -> Z =       545518122089.9991+
         = 148  -> Z =  39660184000219160.0009+
         = 163  -> Z = 262537412640768743.9999999999992+
  ```

  Other such values include also 232, 268, 522, 652, and 719, among infinitely many others, but 163 is still the most amazing of them all.

As for question 4, the solution was already given by Mr. Del Rey. It was a simple curiosity, a coincidence, no explanation necessary or possible, and it didn't require multiprecision at all, anyone could have solved it using any HP model with a solver or a very simple root finding program. I was somewhat surprised that nobody posted the solution much sooner.

Conclusion

Well, it's a pity that no one produced any code to do even basic multiprecision arithmetic in their favorite HP calc, despite the simple guidelines I offered. Anyway, just to show that it isn't as difficult or time-consuming as it seems, I'm including here a short snippet of sample code I wrote for this section. The code is intended to be used as a basis for your own ideas and it isn't optimized at all.

Sample Code

- This sample code is written in the native HP-71B BASIC language and implements a user-defined string function, FNM$, which accepts as input two strings representing multiprecision integer values and returns a string result representing the integer value of their product.

- **Listing:**

```
    10 DEF FNM$[200](A$,B$) @ OPTION BASE 1 @ DIM M,N,L,P,I,J,N1,N2,N3
@ N1=LEN(A$)
    20 N2=LEN(B$) @ N3=N1+N2 @ DIM A(N1),B(N2),C(N3),C$[N3]
    30 FOR I=1 TO N1 @ A(N1+1-I)=VAL(A$[I,I]) @ NEXT I
    40 FOR I=1 TO N2 @ B(N2+1-I)=VAL(B$[I,I]) @ NEXT I
    50 FOR I=1 TO N2 @ M=B(I) @ IF NOT RES THEN 100
    60 L=I @ FOR J=1 TO N1 @ N=A(J) @ IF NOT RES THEN 90
    70 P=M*N @ C(L)=C(L)+RES @ P=RES
    80 IF RES>9 THEN C(L)=RMD(P,10) @ C(L+1)=C(L+1)+P DIV 10
    90 L=L+1 @ NEXT J
```

-
  ```
     100 NEXT I
  ```

-
  ```
     110 FOR I=N3 TO 1 STEP -1 @ IF C(I) THEN 130
  ```

-
  ```
     120 NEXT I
  ```

-
  ```
     130 C$="" @ FOR I=I TO 1 STEP -1 @ C$=C$&STR$(C(I)) @ NEXT I @ IF
  C$="" THEN C$="0"
  ```

-
  ```
     140 FNM$=C$ @ END DEF
  ```
- **Examples:**

-
  ```
         Given:   A$="31415926535897932384"
                  B$="2236067977499"

         We find:

            FNM$(A$,B$) = "70248147310382454885786735427616"
            FNM$(A$,A$) = "986960440108935861844089101446235923456"
            FNM$(B$,B$) = "4999999999996468370295001"

         and to top it all:

            FNM$(FNM$(A$,B$),FNM$(A$,B$)) =

            =
  "4934802200541193730412497902865991365066527558841386130375443456"
  ```
- **Some notes:**

  1. We are representing multiprecision numbers as strings (instead of arrays) and multiprecision operations as user-defined functions. This makes it much more easy for the user to use them interactively, from the command line, no need to write a small BASIC program to execute them.

  2. Line 10 defines FNM$ as a user-defined string function that can return a string up to 200 characters in length. So the maximum result is limited to 200 digits.

  3. The string parameters are meant to represent multiprecision integer values only, not floating point ones. The result will be an integer as well.

  4. The multiprecision integers represented as strings are stored internally in full-precision numeric arrays, one single digit per array element. Then, pairs of single digits are multiplied at a time in a doubly nested loop. This is *extremely* inefficient: the HP-71B can hold integers up to 12 digits long on an array element, so your improved version should store and multiply pairs of 5 or 6 digits at a time. This would increase speed by a factor of 20 or more

  5. Lines 30 and 40 store each digit in their own numeric array element

  6. The commands "IF NOT RES" in lines 50 and 60 skip digits equal to 0. By the way, RES returns the "LASTX" value, i.e.: the last result in a calculation. It's faster than using a variable, and saves memory as well.

  7. Lines 70 and 80 multiply each pair of single digits and take care of the carry if the result exceeds one digit.

  8. Lines 110 and 120 avoid outputting leading zeroes

  9. Line 130 reassembles the result back to a string

Re: S&SMC #4 & #3 Final Remarks...

*Posted by **Andrés C. Rodríguez (Argentina)** on **30 June 2002, 9:06 a.m.**, in response to Re: S&SMC #4: Final Remarks [LONG!] , posted by Ex-PPC member on 29 June 2002, 10:26 p.m.*

Thank you again for an interesting challenge and for the enlightning final remarks. While this time I have had not enough time or interest to participate trying to solve the questions, I think the two-week period is more appropriate for the S&SMCs. Let us hope that the reduced participation in the last episode will not induce Mr. Ex-PPC Member to quit challenging us. It remains to be proved that the participation is larger in odd-numbered S&SMCs than in even-numbered cases. :-)

BTM, Mr. Ex-PPC, while I am fully aware I was late with my final response to S&SMC #3, and also do know that the examples you give in your final remarks were not optimized (so no comparisons are to be made), still I would had liked some comments about my answer (of course, you are not obliged at all to provide such); since I think the program I submitted have a couple of interesting features to discuss. Of course, this could be done by email, but not having your name or email address only left this option for me to voice this.

Of course, I mean no critique or complaint, and please disregard any idiomatic expression which may suggest that.

Short & Sweet Math Challenges for HP fans #5

Posted by ***Ex-PPC member*** *on* ***30 June 2002, 7:31 p.m.***

Last week's challenge dealt with multiprecision computations. It needed some amount of real programming work, so it didn't get as many replies as previous, easier ones. This week's challenge is therefore shorter & sweeter as it needs just a little empirical search, plus simple formula-fitting and evaluation programming and a pinch of theoretical work thrown in for good measure.

Foreword

Imagine that you are a Maths teacher and you are preparing tomorrow's class, which will introduce your students to ***Cardano's formula*** for solving the reduced cubic equation:

```
                x^3 + p*x + q = 0
```

Given ***p*** and ***q***, Cardano's formula gives this value for x:

```
      x = CURT(-q/2+SQRT((q/2)^2+(p/3)^3)) + CURT(-q/2-SQRT((q/2)^2+(p/3)^3))
```

where SQRT means "square root" and CURT means "cube root". Though the formula isn't really particularly complex, you are understandably worried that your students will find it somewhat complicated and weird-looking, so you decide that giving a simple numerical example will make it seem all the more familiar.

With this noble idea in mind, you then try this example, to be developed in the class:

```
                x^3 + 2*x - 7 = 0
```

This corresponds to p = 2 and q = -7, and Cardano's formula gives:

```
      x = CURT(7/2 + SQRT(49/4+8/27)) + CURT(7/2 - SQRT(49/4+8/27))
        = CURT(7/2 + SQRT(1355/108)) + CURT(7/2 - SQRT(1355/108))
        = CURT(7/2 + 3.54207513984) + CURT(7/2 - 3.54207513984)
        = CURT(7.04207513984) + CURT(-0.04207513984)
        = 1.9167562361864 - 0.3478098331340
        = 1.5689464030524
```

which is indeed a root of the equation. Let's check it:

```
   x^3+2*x-7 = 1.5689464030524^3 + 2*1.5689464030524 - 7
             = 3.862107193895 + 3.137892806105 - 7
             = 0
```

Now, you fear understandably that such festival of floating point, many-decimal numbers will not really make the point any simpler, and will probably bore your intended audience, faced with using their calculators extensively in order to find the answer and check it as well. So, you are kept wondering:

```
   "What I really need is to find a simpler example, some suitable values
    for p and q which will make all intermediate and final results small
    integer or rational numbers, so that no calculator shall be necessary
    to perform the computations and the whole process will seem much simpler,
    allowing my students to focus on the formula, not on the computational
    drudgery"
```

So that's what S&SMC#5 is all about:

The Challenge

1. *Use your favorite HP calculator to try and find values of p and q which will make* ***all*** *intermediate and final results in the evaluation of Cardano's formula either exact integers or rational fractions, so that no irrational number ever appears and all computations can be carried out either mentally or simply by hand.*

2. Once you have found a sufficiently large number of solutions (p,q), use your HP calculator to find suitable formulas that will generate them all (polynomial fitting, perhaps ?) or deduce such formulas theoretically. Ideally, the formulas would take as input the desired value for the final root, and would produce p and q such that the resulting cubic equation would have that root and would be extremely simple to solve, as requested.

Recommended HP calculator

- For finding solutions (p,q) any programmable model will do, from the HP-10C or HP-25C onwards. The faster, the merrier, but the programming itself is pretty trivial.

- For fitting the solutions to a polynomial, I guess any model from an HP-11C onwards will be adequate.

Estimated difficulty and allotted time

Pretty easy. Really "short and sweet" this time. Also, you have **two weeks** to try your hand with this challenge. At the end of that period, I'll post the usual ***Final Remarks***, including solutions and snippets of code if necessary. If the number of postings in this thread warrants it, I will then post the next challenge, S&SMC#6.

By the way, I can't resist: if any of you finally developed some multiprecision routines, you can test that the root of the above equation, to 77 decimals places, is:

```
x =
1.5689464030523822673523347516877514055016871136518810379294694517065543130222
72
```

A question to you, Valentin Albillo (Ex-PPC), regarding your "challenges"

Posted by ***W. Bruce Maguire II*** *on* ***3 July 2002, 2:12 p.m.****, in response to Short & Sweet Math Challenges for HP fans #5, posted by Ex-PPC member on 30 June 2002, 7:31 p.m.*

Dear Valentin:
(*aka* Ex-PPC Member)
(*aka* Ex-PPC #4747)
(*aka* HPCC #1075)

I'm curious Valentin; out of the population of regular viewers/participators in the MoHPC forum, how many do you actually *believe* have so much free time on their hands that they want someone to give them math/calclator-related homework?

After reading all your "challenges" up to this one (number *five*), I sense a strong air of condescension. I guess others who have responded in the past don't feel this way, or don't care, but I must say that I'm continually amazed. Your "challenges" are *not* the light-hearted, friendly kind (like "who can generate the number 100 in the fewest keystrokes without hitting a number key?"); rather, they are the instructor/pupil kind of challenges---where Valentin is the instructor and we are the pupils.

It seems to me that your "challenges" serve only *you*. Who knows how many weeks/months/years you spend on some of these trivial pursuits, before you issue your challenge and smugly enjoy the "inferior" efforts produced by your "pupils"?

I know that there is *absolutely* no way for me to voice my disgust with the tone of your "challenges" without sounding petty. And you certainly have *every* right to continue to post your "challenges"---I am certainly no moderator for this forum---but I wanted to let you know that there is at least *one* person on this list who is "smart" enough (in your eyes maybe) to realize the purpose of your tests. It is clear to this non-pupil that your "challenges" are wholly self-serving. I, personally, am far too busy to indulge your ego.

I have a question for any HP-fan who *has* been responding to Valentin's "challenges". I have to ask: are you not able to challenge *yourself*? If you enjoy this type of pursuit (I, myself, have been known to tilt at a few mathematical windmills with my HPs! ;-)), then I would submit that there is an overwhelming amount of material available in books and on the Internet that will provide you with enough challenges for a lifetime. And you can pursue *these* challeges on your own timeframe and not suffer the supercilious responses of the self-satisfied Valentin Albillo.

And finally, Valentin, I would assert that your contributions to this forum are miniscule compared with what they *could* be. You haven't posted anything in the "Articles" section; why? I present your posts about trigonometric functions on the HP-12C as an example. You asked for information from others, posted patronizing responses to anyone who offered solutions, and refused to *ever* respond to multiple requests that you post the code that *you* developed. A subsequent issue of the HPCC Datafile exposed *why* you wouldn't respond to anyone asking to see your programs: you had to make sure that your work was *in print* before revealing it to anyone! You clearly have the time and ability to do good things with HP calculators. If you cared less about elevating your self-image, *many* might benefit from your work.

For any who have access to the (excellent!) HPCC Datafile publications, I invite you to read the Editor's note at the bottom of page 20 of the Jan/Feb 2002 issue (V21,N1) and the first paragraph of Wlodek's article on page 16 of the Mar/Apr 2002 issue (V21,N2). They unintentionally betray how Valentin feels about his "teachings". That is, the sharing of knowledge is not what is important; what is important is that Valentin takes center-stage!

Bruce.
(I may be wrong, but at least I have the courage to post with my NAME and my E-MAIL address!)

An answer

*Posted by **Nenad Vulic (Croatia)** on **4 July 2002, 3:23 a.m.**, in response to A question to you, Valentin Albillo (Ex-PPC), regarding your "challenges", posted by W. Bruce Maguire II on 3 July 2002, 2:12 p.m.*

Bruce wrote:

"...who can generate the number 100 in the fewest keystrokes without hitting a number key?"

CLx, 10^x, 10^x, ENTER, X

I did not feel smart enough for the previous challenges, but I love this one. It took only a few seconds of my time...

---

and with a 12C...

*Posted by **thibaut.be** on **4 July 2002, 4:20 a.m.**, in response to An answer, posted by Nenad Vulic (Croatia) on 4 July 2002, 3:23 a.m.*

CLX e^x ENTER e^x g INTG D %

;-) I also enjoyed this one... LOL !

---

shorter abswer for HP 12C

*Posted by **mapet** on **4 July 2002, 5:05 a.m.**, in response to and with a 12C..., posted by thibaut.be on 4 July 2002, 4:20 a.m.*

CLX e^x ENTER %T - 4 keystrokes

OVER

Re: shorter abswer for HP 12C

*Posted by **thibaut.be** on **4 July 2002, 5:23 a.m.**, in response to shorter abswer for HP 12C, posted by mapet on 4 July 2002, 5:05 a.m.*

yes, indeed, ;-)

alternative anserws

*Posted by **mapet** on **4 July 2002, 5:26 a.m.**, in response to shorter abswer for HP 12C, posted by mapet on 4 July 2002, 5:05 a.m.*

ENTER EEX ENTER %T

and my favourite but longer one:

CLX EEX ENTER % 1/x

OVER

---

and the best answer....

*Posted by **mapet** on **4 July 2002, 5:30 a.m.**, in response to alternative anserws, posted by mapet on 4 July 2002, 5:26 a.m.*

EEX ENTER %T

OVER :-) LOL

Re: and the best answer....

*Posted by **Thibaut** on **4 July 2002, 9:51 a.m.**, in response to and the best answer...., posted by mapet on 4 July 2002, 5:30 a.m.*

... or

S+ ENTER %T

;-)

Re: A slightly-better answer

*Posted by **Karl Schneider** on **4 July 2002, 4:20 a.m.**, in response to An answer, posted by Nenad Vulic (Croatia) on 4 July 2002, 3:23 a.m.*

Nenad --

Your technique was the same one I had in mind, except that x^2 could be substituted for ENTER, * (except if you're using a 12C):

CLx, 10^x, 10^x, x^2

4 operations; total number of keystrokes dependent on calculator model...

Some shorter options, but some shortcuts too...

*Posted by **Andrés C. Rodríguez (Argentina)** on **4 July 2002, 7:52 a.m.**, in response to Re: A slightly-better answer, posted by Karl Schneider on 4 July 2002, 4:20 a.m.*

[EEX] [10^X] (2 keystrokes)

or

[ACOS] (1 keystroke, assuming GRAD mode and a cleared machine)

or

[R/S] (1 keystroke, calling a one line program: 01 1E2)

or

[ON] (1 keystroke, on a Continuous Memory calculator which displayed 100 when turned off)

or

A wand scan of a "100" data sticker (1 button press and a handstroke)

or ...

The consequence

*Posted by **Nenad Vulic (Croatia)** on **4 July 2002, 8:22 a.m.**, in response to Some shorter options, but some shortcuts too..., posted by Andrés C. Rodríguez (Argentina) on 4 July 2002, 7:52 a.m.*

... is that the number of responses in this thread proves that Bruce was obviously right.

Re: Some shorter options, but some shortcuts too...

*Posted by **thibaut.be** on **4 July 2002, 9:50 a.m.**, in response to Some shorter options, but some shortcuts too..., posted by Andrés C. Rodríguez (Argentina) on 4 July 2002, 7:52 a.m.*

You're cheeting, you're not using a 12c ;-)

Sorry, no 12C here

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***4 July 2002, 10:22 a.m.****, in response to Re: Some shorter options, but some shortcuts too..., posted by thibaut.be on 4 July 2002, 9:50 a.m.*

Yes, I have no 12C at hand. And I was not taking it too seriously, as you surely noticed.

Correction

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***4 July 2002, 10:25 a.m.****, in response to Some shorter options, but some shortcuts too..., posted by Andrés C. Rodríguez (Argentina) on 4 July 2002, 7:52 a.m.*

First example has a [x^2] missing, so it is 3 keystrokes.

Please see: Me. Bruce's challenge didn't specified any particular model (12C or whatever)

Re: Correction

Posted by ***John Ioannidis*** *on* ***5 July 2002, 11:43 a.m.****, in response to Correction, posted by Andrés C. Rodríguez (Argentina) on 4 July 2002, 10:25 a.m.*

I think the arccosine of zero in grad mode wins the originality contest, even though it is five keystrokes on most handheld HPs: [shift] [GRD] [CLx] [shift] [ACOS]. Congratulations Andrés!

I would count [EEX] and [.] as numeric keys.

Thank you! (GRAD ACOS)

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***5 July 2002, 12:20 p.m.****, in response to Re: Correction, posted by John Ioannidis on 5 July 2002, 11:43 a.m.*

Thank you, John, for the originality nomination. Around 1977 I needed to program my HP 25 to "bounce" some values at the 100 boundary; my function needed to work this way:

[x always had a value between 0 and 200]

If x < 100, then f(x) = x

If x > 100, then f(x) = 100 - x

By using GRAD mode (the function had nothing to do with angles), it was very convenient to execute:

SIN ASIN

and that was all about it!

I suppose the faint and very old memory of that case influenced my answer...

Re: A question to you, Valentin Albillo (Ex-PPC), regarding your "challenges"

*Posted by **Vieira, Luiz C. (Brazil)** on **4 July 2002, 8:34 a.m.**, in response to A question to you, Valentin Albillo (Ex-PPC), regarding your "challenges", posted by W. Bruce Maguire II on 3 July 2002, 2:12 p.m.*

Hi, Mr. Bruce

I intend no harm, and if I offend someone, please, believe this is not the purpose.

I´d like to express my concerns about the way people show their thoughts: maybe the way I express myself seems offensive to someone, but it may also be fairly accepted in my own community. That's why I sometimes think a lot before writing anything in here, mostly because I'm a foreigner and I may not understand the real meaning of some idiomatic expressions, but some words speak for themselves.

I felt (my own opinion, please) sometimes that Mr. Ex-PPC Member presented the challenges in a way not compatible with this forum's way. In some (many) cases, the single mentioning of any HP calc as a wonderful, supreme or-the-like instrument, made me suspect about the real concern: solving the problem OR pointing HP calcs as not enough for the job.

I solved one or two challenges for my own, but I didn't think of them, the way they were shown, with the purpose to serve this forum's primary needs. I suspect Mr. Ex-PPC member (as a super-hero with his identity revealed) may disappear for a while OR change his identity again, I don't know.

I agree with Mr. Maguire´s post primary concerns, **and I don't want my friends in here to understand my own post as a negative criticism to someone else´s,** only another view of his mention.

Cheers.

Re: A question to you, W. Bruce Maguire II

*Posted by **Ellis Easley** on **8 July 2002, 9:03 a.m.**, in response to A question to you, Valentin Albillo (Ex-PPC), regarding your "challenges", posted by W. Bruce Maguire II on 3 July 2002, 2:12 p.m.*

As they say in talk radio, "I love you, man", but while Ex-PPC might be pretentious, you are a trouble maker. There's no harm in Ex-PPC's challenges. And regarding the use of aliases, will you say unequivocally that you are not "John Smith", the author of this message:

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/forum.cgi?read=19556

Re: A question to you, W. Bruce Maguire II

*Posted by **W. Bruce Maguire II** on **8 July 2002, 7:13 p.m.**, in response to Re: A question to you, W. Bruce Maguire II, posted by Ellis Easley on 8 July 2002, 9:03 a.m.*

Ellis:

You wrote:

As they say in talk radio, "I love you, man", but while Ex-PPC might be pretentious, you are a trouble maker. There's no harm in Ex-PPC's challenges. And regarding the use of aliases, will you say unequivocally that you are not "John Smith", the author of this message:

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/forum.cgi?read=19556

Well, I guess it depends on what you mean by troublemaker... ;-) And actually, I thought there was a very small amount of harm in Valentin's (Ex-PPC's) posts---they are so downright condescending that I thought they slightly harmed *everyone* in the forum. You, and possibly others, may disagree with that notion. That's OK with me. The message really *was* to Valentin; although he never responded to my questions before, so I didn't really expect a response this time.

As to the alias and the above message, I will *unequivocally* state that I did *NOT* post that message. I'll go one better: I give you my word. I have *never* posted here using any alias. Mike *has* accused me of trying to fool people by using my TWO e-mail addresses! He caught me! ;-) I didn't think *anyone* would figure out that "W. Bruce Maguire II, maguire@AnalyticInvestments.com" and "W. Bruce Maguire II, maguire@ti.com" were the same... That Mike's a clever one!

I *do* admit to silently laughing when I saw the post, though. Believe it or not, there are other people present who don't like Mike's business practices either.

I've never felt that I had anything to gain by posting anonymously, or anything to lose by using my real name and e-mail address. I'm curious why Valentin (Ex-PPC) feels differently.

Bruce.

Re: A question to you, W. Bruce Maguire II

*Posted by **Ellis Easley** on **9 July 2002, 3:34 a.m.**, in response to Re: A question to you, W. Bruce Maguire II, posted by W. Bruce Maguire II on 8 July 2002, 7:13 p.m.*

"John Smith"'s remark about not accepting PayPal made me think of you - maybe I've got a screw loose!

Re: A question to you, W. Bruce Maguire II

*Posted by **W. Bruce Maguire II** on **9 July 2002, 12:28 p.m.**, in response to Re: A question to you, W. Bruce Maguire II, posted by Ellis Easley on 9 July 2002, 3:34 a.m.*

"John Smith"'s remark about not accepting PayPal made me think of you - maybe I've got a screw loose!

Ellis:

No, you don't have a screw loose---at least I don't think you do... ;-) That is the *exact* same complaint that *I* had about Mike's business. But, as I said, there are others around that have privately voiced to me (via e-mail) similar views regarding Mike's business.

Speaking of a screw loose, I still need to check-up on HP screws at McGuckin's for someone (I'll have to check the archives to see who I owe that to)! It's not you, is it? If it is, then I appreciate the subtle reminder! ;-) To be honest, I've had precious little time to devote to all-things-HP lately...

Bruce.

Re: A question to you, W. Bruce Maguire II

*Posted by **Ellis Easley** on **10 July 2002, 11:06 a.m.**, in response to Re: A question to you, W. Bruce Maguire II, posted by W. Bruce Maguire II on 9 July 2002, 12:28 p.m.*

I've learned a lot about screws since that discussion started. I now have a number of different #2 and #3 screws including a #2 Plastite, none of which are quite right. Currently I believe the original screw was a "Plastite 60-1" and while that is obsolete, and the most widely made Plastite screw now is "Plastite 48-2", at least one manufacturer offers a "Plastite 48-1", which I think will fit. I need to make some microscope observations before I ask a dealer to try to get some.

Here is a link to the original thread: http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv008.cgi?read=17614

And my three updates: http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv008.cgi?read=17963

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv008.cgi?read=18432

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv008.cgi?read=19079

Re: Short & Sweet Math Challenges for HP fans #5

*Posted by **Rupert (Northern Italy, EU)** on **4 July 2002, 4:17 p.m.**, in response to Short & Sweet Math Challenges for HP fans #5, posted by Ex-PPC member on 30 June 2002, 7:31 p.m.*

>
> x^3 + p*x + q = 0
>
> Given p and q, Cardano's formula gives this value for x:
>
>
> x = CURT(-q/2+SQRT((q/2)^2+(p/3)^3)) + CURT(-q/2-SQRT((q/2)^2+(p/3)^3))
>

Well, if I'm not wrong, the signs in the second term are a bit
messed up. The correct version should be rather:

x = CURT(-q / 2 + SQR((q / 2) ^ 2 + (p / 3) ^ 3)) - CURT(q / 2 + SQR((q / 2) ^ 2 + (p / 3) ^ 3))

>
> x^3 + 2*x - 7 = 0
>
> This corresponds to p = 2 and q = -7, and Cardano's formula gives:
>
>
> x = CURT(7/2 + SQRT(49/4+8/27)) + CURT(7/2 - SQRT(49/4+8/27))
> = CURT(7/2 + SQRT(1355/108)) + CURT(7/2 - SQRT(1355/108))
> = CURT(7/2 + 3.54207513984) + CURT(7/2 - 3.54207513984)
> = CURT(7.04207513984) + CURT(-0.04207513984)
> = 1.9167562361864 - 0.3478098331340
> = 1.5689464030524
>

Again, the final result is right (I checked it with the HP48G+ ROOT
finder :)) but the signs in your intermediate passages are all
messed up: CURT(-0.04207513984) would give a complex result.

Anyway, if I haven't misunderstood what is being asked, it looks
too easy: I just imposed that the fractional parts of the two
terms of the Cardano's formula and of the root must be equal to
zero. There is plenty of results: I enclose below the
quick'n'dirty :) BASIC listing I wrote to generate them:

```
DIM d, t1, t2, x AS DOUBLE
DIM p, q, kp, kq, maxnum AS DOUBLE

maxnum = 1000
```

```
OPEN "c:\outfile.dat" FOR OUTPUT AS #1

CLS

FOR p = 1 TO maxnum
kp = (p / 3) ^ 3
FOR q = -1 TO -maxnum STEP -1
kq = q / 2
d = SQR(kq ^ 2 + kp)
t1 = -kq + d
t2 = kq + d
x = t1 ^ (1 / 3) - t2 ^ (1 / 3)

IF t1 - INT(t1) = 0 AND t2 - INT(t2) = 0 AND x - INT(x) = 0 THEN

PRINT "p="; p; "q="; q; "d="; d; "t1="; t1; "t2="; t2; "x="; x

PRINT #1, "p="; p; "q="; q; "d="; d; "t1="; t1; "t2="; t2; "x="; x

END IF

NEXT q
NEXT p

CLOSE #1

END
```

I haven't had much time, so I tested the algorithm in BASIC and used my HP48G+ only to check the results. Here is a sample of the output:

```
p= 6 q=-7 d= 4.5 t1= 8 t2= 1 x= 1
p= 9 q=-26 d= 14 t1= 27 t2= 1 x= 2
p= 12 q=-63 d= 32.5 t1= 64 t2= 1 x= 3
p= 15 q=-124 d= 63 t1= 125 t2= 1 x= 4
p= 18 q=-19 d= 17.5 t1= 27 t2= 8 x= 1
p= 18 q=-215 d= 108.5 t1= 216 t2= 1 x= 5
p= 21 q=-342 d= 172 t1= 343 t2= 1 x= 6
p= 24 q=-56 d= 36 t1= 64 t2= 8 x= 2
p= 24 q=-511 d= 256.5 t1= 512 t2= 1 x= 7
p= 27 q=-728 d= 365 t1= 729 t2= 1 x= 8
p= 30 q=-117 d= 66.5 t1= 125 t2= 8 x= 3
p= 30 q=-999 d= 500.5 t1= 1000 t2= 1 x= 9
p= 36 q=-37 d= 45.5 t1= 64 t2= 27 x= 1
p= 36 q=-208 d= 112 t1= 216 t2= 8 x= 4
p= 42 q=-335 d= 175.5 t1= 343 t2= 8 x= 5
p= 45 q=-98 d= 76 t1= 125 t2= 27 x= 2
p= 48 q=-504 d= 260 t1= 512 t2= 8 x= 6
p= 54 q=-189 d= 121.5 t1= 216 t2= 27 x= 3
```

```
p= 54 q=-721 d= 368.5 t1= 729 t2= 8 x= 7
p= 60 q=-61 d= 94.5 t1= 125 t2= 64 x= 1
p= 60 q=-992 d= 504 t1= 1000 t2= 8 x= 8
p= 63 q=-316 d= 185 t1= 343 t2= 27 x= 4
p= 72 q=-152 d= 140 t1= 216 t2= 64 x= 2
p= 72 q=-485 d= 269.5 t1= 512 t2= 27 x= 5
p= 81 q=-702 d= 378 t1= 729 t2= 27 x= 6
p= 84 q=-279 d= 203.5 t1= 343 t2= 64 x= 3
p= 90 q=-91 d= 170.5 t1= 216 t2= 125 x= 1
p= 90 q=-973 d= 513.5 t1= 1000 t2= 27 x= 7
p= 96 q=-448 d= 288 t1= 512 t2= 64 x= 4
p= 105 q=-218 d= 234 t1= 343 t2= 125 x= 2
p= 108 q=-665 d= 396.5 t1= 729 t2= 64 x= 5
p= 120 q=-387 d= 318.5 t1= 512 t2= 125 x= 3
p= 120 q=-936 d= 532 t1= 1000 t2= 64 x= 6
p= 126 q=-127 d= 279.5 t1= 343 t2= 216 x= 1
p= 135 q=-604 d= 427 t1= 729 t2= 125 x= 4
p= 144 q=-296 d= 364 t1= 512 t2= 216 x= 2
p= 150 q=-875 d= 562.5 t1= 1000 t2= 125 x= 5
[...]
```

The search is limited to 1000 (maxnum) 'cause there are too much results,
anyway it's easy to modify that limit in order to go further.
Turning the line "FOR q = -1 TO -maxnum STEP -1" into "FOR q = 1 TO maxnum STEP 1",
I found that, if a couple (p,q) is a valid result and gives a root x, then (p,-q) is still a
valid result and gives a root -x, too.

If that is what it's being asked, maybe I'll find the time to go ahead with
polynomial interpolation of the results in order to have some fun with my HP48. :)

--

Re: Short & Sweet Math Challenges for HP fans #5

*Posted by **John Ioannidis** on **5 July 2002, 12:24 a.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #5, posted by Rupert (Northern Italy, EU) on 4 July 2002, 4:17 p.m.*

You are wrong on so many counts.

1. cbrt(-x) = -cbrt(x), so the signs are correct.

2. The cube root of a real is a real, regardless of whether that real is positive or negative. (Yes, I know about the other two complex roots, but that's really (no pun) for a complex number with a zero imaginary part. Think algebraic fields).

3. Your program missed the obvious solution p=-6, q=4 (with root 2). No wonder, since...

4. Computers (and calculators) do arithmetic with finite precision; your test "t1 - int(t1)" is bound to fail when t1 should be an integer but isn't because of rounding errors.

Re: Short & Sweet Math Challenges for HP fans #5

*Posted by **Rupert (Northern Italy, EU)** on **6 July 2002, 8:12 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #5, posted by John Ioannidis on 5 July 2002, 12:24 a.m.*

> You are wrong on so many counts.
>
> 1. cbrt(-x) = -cbrt(x), so the signs are correct.
>
> 2. The cube root of a real is a real, regardless of whether that real is positive or negative. (Yes, I know about the other two complex roots, but that's really (no pun) for a complex number with a zero imaginary part. Think algebraic fields).
>

Here is an example: (-8)^(1/3).

MATHCAD: 1+1.732050807568877*i

HP48G+: (1, 1.73205080757)

HP32SII: x:1 y:1.732050808

etc.

> 3. Your program missed the obvious solution p=-6, q=4 (with root 2).

Well, I didn't even search for negative values of p.
Since it's the weekend :-) I took the time to look for
the right version of the Cardano's formula for the cubic equation
x^3+P*x+Q=0 and found the following:

x=((-Q+SQR(Q^2+4*P^3/27))/2)^(1/3) + ((-Q-SQR(Q^2+4*P^3/27))/2)^(1/3)

Now my new quick'n'dirty program tests all the couples (p,q), (-p,q),
(p,-q), and (-p,-q), too.

```
DECLARE FUNCTION test (dummy$)

DIM p, q, r AS DOUBLE
DIM SHARED p0, q0, r0, e, eps, epsx AS DOUBLE
DIM maxnum, k, k1p, q2, dummy AS DOUBLE

k = 4 / 27
e = 1 / 3
eps = .000001
epsx = .0001
```

```
CLS
PRINT ""
INPUT "Max p, q "; maxnum

OPEN "c:\outfile.dat" FOR OUTPUT AS #1

FOR p = 1 TO maxnum
k1p = k * (p ^ 3)

FOR q = 1 TO maxnum
q2 = q ^ 2
r0 = q2 + k1p
p0 = p
q0 = q
dummy = test("")

q0 = -q
dummy = test("")

r0 = q2 - k1p
p0 = -p
q0 = q
dummy = test("")

q0 = -q
dummy = test("")

NEXT q
NEXT p

CLOSE #1

END

FUNCTION test (dummy$)
DIM sr, t1, t2, x AS DOUBLE

IF r0 >= 0 THEN
sr = SQR(r0)
IF ABS(sr - INT(sr)) <= eps THEN
sr = INT(sr + .5)
t1 = (-q0 + sr)
t2 = (-q0 - sr)

IF t1 >= 0 AND t2 >= 0 THEN
x = (t1 / 2) ^ e + (t2 / 2) ^ e
IF ABS(x - INT(x)) <= epsx THEN
x = INT(x + .5)
PRINT "p="; p0; "q="; q0; "discr.="; r0; "t1="; t1; "t2="; t2; "x="; x
PRINT #1, "p="; p0; "q="; q0; "discr.="; r0; "t1="; t1; "t2="; t2; "x="; x
END IF
END IF
END IF
END IF

test = 0
END FUNCTION
```

Output with search limited to a maximum absolute value of 1000 for both p and q:

```
p=-3 q=-2 discr.= 0 t1= 2 t2= 2 x= 2
p=-6 q=-9 discr.= 49 t1= 16 t2= 2 x= 3
p=-9 q=-28 discr.= 676 t1= 54 t2= 2 x= 4
p=-12 q=-16 discr.= 0 t1= 16 t2= 16 x= 4
p=-12 q=-65 discr.= 3969 t1= 128 t2= 2 x= 5
p=-15 q=-126 discr.= 15376 t1= 250 t2= 2 x= 6
p=-18 q=-35 discr.= 361 t1= 54 t2= 16 x= 5
p=-18 q=-217 discr.= 46225 t1= 432 t2= 2 x= 7
p=-21 q=-344 discr.= 116964 t1= 686 t2= 2 x= 8
p=-24 q=-72 discr.= 3136 t1= 128 t2= 16 x= 6
p=-24 q=-513 discr.= 261121 t1= 1024 t2= 2 x= 9
p=-27 q=-54 discr.= 0 t1= 54 t2= 54 x= 6
p=-27 q=-730 discr.= 529984 t1= 1458 t2= 2 x= 10
p=-30 q=-133 discr.= 13689 t1= 250 t2= 16 x= 7
p=-36 q=-91 discr.= 1369 t1= 128 t2= 54 x= 7
p=-36 q=-224 discr.= 43264 t1= 432 t2= 16 x= 8
p=-42 q=-351 discr.= 112225 t1= 686 t2= 16 x= 9
p=-45 q=-152 discr.= 9604 t1= 250 t2= 54 x= 8
p=-48 q=-128 discr.= 0 t1= 128 t2= 128 x= 8
p=-48 q=-520 discr.= 254016 t1= 1024 t2= 16 x= 10
p=-54 q=-243 discr.= 35721 t1= 432 t2= 54 x= 9
p=-54 q=-737 discr.= 519841 t1= 1458 t2= 16 x= 11
p=-60 q=-189 discr.= 3721 t1= 250 t2= 128 x= 9
p=-63 q=-370 discr.= 99856 t1= 686 t2= 54 x= 10
p=-72 q=-280 discr.= 23104 t1= 432 t2= 128 x= 10
p=-72 q=-539 discr.= 235225 t1= 1024 t2= 54 x= 11
p=-75 q=-250 discr.= 0 t1= 250 t2= 250 x= 10
p=-81 q=-756 discr.= 492804 t1= 1458 t2= 54 x= 12
p=-84 q=-407 discr.= 77841 t1= 686 t2= 128 x= 11
p=-90 q=-341 discr.= 8281 t1= 432 t2= 250 x= 11
p=-96 q=-576 discr.= 200704 t1= 1024 t2= 128 x= 12
p=-105 q=-468 discr.= 47524 t1= 686 t2= 250 x= 12
p=-108 q=-432 discr.= 0 t1= 432 t2= 432 x= 12
p=-108 q=-793 discr.= 442225 t1= 1458 t2= 128 x= 13
p=-120 q=-637 discr.= 149769 t1= 1024 t2= 250 x= 13
p=-126 q=-559 discr.= 16129 t1= 686 t2= 432 x= 13
p=-135 q=-854 discr.= 364816 t1= 1458 t2= 250 x= 14
p=-144 q=-728 discr.= 87616 t1= 1024 t2= 432 x= 14
p=-147 q=-686 discr.= 0 t1= 686 t2= 686 x= 14
p=-162 q=-945 discr.= 263169 t1= 1458 t2= 432 x= 15
```

```
p=-168 q=-855 discr.= 28561 t1= 1024 t2= 686 x= 15
```

I got results with p>0 only doing a search with a maximum absolute
value > 1000 for both p and q, but results with p<0 and q>0
(as (-6,4)) are still missing:

```
p= 1 q=-1331 discr.= 1771561 t1= 2662 t2= 0 x= 11
p=-1 q=-1331 discr.= 1771561 t1= 2662 t2= 0 x= 11
p= 1 q=-2197 discr.= 4826809 t1= 4394 t2= 0 x= 13
p=-1 q=-2197 discr.= 4826809 t1= 4394 t2= 0 x= 13
p= 1 q=-2744 discr.= 7529536 t1= 5488 t2= 0 x= 14
p=-1 q=-2744 discr.= 7529536 t1= 5488 t2= 0 x= 14
p= 1 q=-3375 discr.= 1.139063E+07 t1= 6750 t2= 0 x= 15
p=-1 q=-3375 discr.= 1.139063E+07 t1= 6750 t2= 0 x= 15
p= 1 q=-4096 discr.= 1.677722E+07 t1= 8192 t2= 0 x= 16
p=-1 q=-4096 discr.= 1.677722E+07 t1= 8192 t2= 0 x= 16
p= 1 q=-4913 discr.= 2.413757E+07 t1= 9826 t2= 0 x= 17
p=-1 q=-4913 discr.= 2.413757E+07 t1= 9826 t2= 0 x= 17
p= 1 q=-5832 discr.= 3.401222E+07 t1= 11664 t2= 0 x= 18
p=-1 q=-5832 discr.= 3.401222E+07 t1= 11664 t2= 0 x= 18
p= 1 q=-6859 discr.= 4.704588E+07 t1= 13718 t2= 0 x= 19
p=-1 q=-6859 discr.= 4.704588E+07 t1= 13718 t2= 0 x= 19
p= 1 q=-8000 discr.= 6.4E+07 t1= 16000 t2= 0 x= 20
p=-1 q=-8000 discr.= 6.4E+07 t1= 16000 t2= 0 x= 20
p= 1 q=-9261 discr.= 8.576612E+07 t1= 18522 t2= 0 x= 21
p=-1 q=-9261 discr.= 8.576612E+07 t1= 18522 t2= 0 x= 21
p= 2 q=-4096 discr.= 1.677722E+07 t1= 8192 t2= 0 x= 16
p= 2 q=-4913 discr.= 2.413757E+07 t1= 9826 t2= 0 x= 17
p= 2 q=-5832 discr.= 3.401222E+07 t1= 11664 t2= 0 x= 18
p=-2 q=-5832 discr.= 3.401222E+07 t1= 11664 t2= 0 x= 18
p= 2 q=-6859 discr.= 4.704588E+07 t1= 13718 t2= 0 x= 19
p=-2 q=-6859 discr.= 4.704588E+07 t1= 13718 t2= 0 x= 19
p= 2 q=-8000 discr.= 6.4E+07 t1= 16000 t2= 0 x= 20
p=-2 q=-8000 discr.= 6.4E+07 t1= 16000 t2= 0 x= 20
p= 2 q=-9261 discr.= 8.576612E+07 t1= 18522 t2= 0 x= 21
p=-2 q=-9261 discr.= 8.576612E+07 t1= 18522 t2= 0 x= 21
p=-3 q=-2 discr.= 0 t1= 2 t2= 2 x= 2
p= 3 q=-6859 discr.= 4.704588E+07 t1= 13718 t2= 0 x= 19
p= 3 q=-9261 discr.= 8.576613E+07 t1= 18522 t2= 0 x= 21
p=-3 q=-9261 discr.= 8.576611E+07 t1= 18522 t2= 0 x= 21
p= 4 q=-9261 discr.= 8.576613E+07 t1= 18522 t2= 0 x= 21
p=-4 q=-9261 discr.= 8.576611E+07 t1= 18522 t2= 0 x= 21
p=-6 q=-9 discr.= 49 t1= 16 t2= 2 x= 3
p=-9 q=-28 discr.= 676 t1= 54 t2= 2 x= 4
[...]
```

> No wonder, since...
>
> 4. Computers (and calculators) do arithmetic with finite precision;
> your test "t1 - int(t1)" is bound to fail when t1 should be an integer

> but isn't because of rounding errors.

Now I have introduced a tolerance in the tests, but some results
are still missing.

Well, I' didn't resist the temptation to give it a try,
but I'm not a mathematician nor a programmer.
From a pratical point of view, any result that get the job done is good
Doing a technical job, when I run into an equation I just solve it...

--

Re: Short & Sweet Math Challenges for HP fans #5

*Posted by **John Ioannidis** on **7 July 2002, 12:20 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #5, posted by Rupert (Northern Italy, EU) on 6 July 2002, 8:12 p.m.*

> Here is an example: (-8)^(1/3).

Is it too much to hope that people first use their brains and then their calculators? (-2)^3 = -8, therefore (-8)^(1/3) = -2.

I did say that you should only think in terms of real numbers. If you work with complex numbers, then -8 is really (-8, 0) (or -8+0j), which will indeed have three cube roots: (-2, 0), (1, sqrt(3)), and (1, -sqrt(3)). If you raise any of these three complex numbers to the third power, you get (-8, 0).

That's high school math, it's not like it's complicated!

The order in which you actually do the arithmetic in order to calcualate the roots may require you to apply the distributive property of multiplication and take the negative sign outside the second cube root to get your calculator to compute the real cube root rather than one of the complex cube roots. But this doesn't mean that the formula is incorrect.

Re: Short & Sweet Math Challenges for HP fans #5

*Posted by **Alex Binca** on **7 July 2002, 3:02 p.m.**, in response to Re: Short & Sweet Math Challenges for HP fans #5, posted by John Ioannidis on 7 July 2002, 12:20 p.m.*

> That's high school math, it's not like it's complicated!

Hey, why don't you try to be a little nicer. Being nice to other people it's taught in kindergarten. It's not like it's complicated!

:) :)

-- Alex Binca

Re: Short & Sweet Math Challenges for HP fans #5

Posted by ***Rupert (Northern Italy, EU)*** *on* ***12 July 2002, 6:41 p.m.****, in response to Re: Short & Sweet Math Challenges for HP fans #5, posted by Rupert (Northern Italy, EU) on 6 July 2002, 8:12 p.m.*

Judging by some replies I've got since now,
it looks like I've been talking in Greek language.
The point is that, using either a CAS or a calculator,
the result for the cubic root of a negative real number
will always be a complex number, or an error message. ;-))
Just this.
Once again, as a little example here are the results
for (-8)^(1/3) :

Mathcad : 1+1.732050807568877*i

HP48G+ : (1, 1.73205080757)

HP32SII : x: 1 y: 1.732050808

HP15C : Re: 1 Im: 1.732050807

```
HP41CX + Math I Pac: Z^1/N, with N=3: U=1  V=1.732050808
                                      U=-2 V=0
                                      U=1  V=-1.732050808
                     Z^N, with N=1/3: U=1  V=1.732050807


Yorkem (HP49G emulator): 2*((1+i*SQRT(3))/2)=1+i*SQRT(3)=
                         =(1, 1.73205080757).
```

It looks like the good ol' 41CX is the overall winner,
being the only one that gives the full result, satisfying
also those peoples who pretend (-k)^(1/3) is just
equal to -(k^(1/3)).

Well, now back on topic.
Since the cubic equation is just a particular case of
a third grade equation, then its roots can be computed
using polynomial root finding algorithms rather than
the Cardano's formula, overriding the troubles it involves.

Here is a new quick'n'dirty listing for the HP48 based
upon the PROOT (Polynomial ROOT) command
(tried also ROOT together with a deflation routine,
but it's obviously slower).
The program just discards those values of P and Q that
generate complex roots or real but non-integer roots.

```
-----------------------------------------------------------------

var: RUN

<< 0 FIX 1 CF
   1E-5 'EPS' STO

   "ENTER LIMIT"
   { ":MAX P,Q: "
   {1 0} V}
   INPUT OBJ->
   -> MXN <<

   1 MXN FOR PL
     1 MXN FOR QL

     PL     QL     TESTPQ
     PL     QL NEG TESTPQ
     PL NEG QL     TESTPQ
     PL NEG QL NEG TESTPQ

     NEXT
   NEXT
>>

1000 .5 BEEP
>>


var: TESTPQ

<< 'Q' STO 'P' STO
   1 0 P Q 4 ->ARRY
   PROOT -> ROOTS <<
   1 3 FOR J
     ROOTS J GET TESTX
   NEXT
>>
>>


var: TESTX

<< -> X <<
   'X' VTYPE
    -> VT <<
   IF VT 0 == THEN
     IF X FP ABS EPS <= THEN

       IF 1 FS? THEN
         P Q X 3 ->ARRY
         -> PQX << RSLT ARRY->
         OBJ-> DROP DROP 1 +
         RSLT SWAP
         PQX SWAP ROW+ >>
       ELSE
          1 SF
          P Q X 1 3 2 ->LIST
          ->ARRY
       END

       'RSLT' STO
       "P=" P +
       "Q=" Q +
       "X=" X +
       CLLCD
       3 DISP
       2 DISP
       1 DISP
       3 FREEZE
     END
   END
>>
>>
>>
```

Results are stored in the matrix RSLT.
Here is some results:

RSLT:

```
------------------
P Q X
------------------

-2 1 1
-2 -1 -1
-3 2 1
-3 2 -2
-3 -2 -1
-3 -2 2
-4 3 1
-4 -3 -1
-5 2 2
-5 -2 -2
-5 4 1
-5 -4 -1
-6 4 2
-6 -4 -2
-6 5 1
-6 -5 -1
-7 6 1
-7 6 2
-7 6 -3
-7 -6 -1
-7 -6 -2
-7 -6 3
-8 3 -3
-8 -3 3
-8 7 1
-8 -7 -1
-8 8 2
-8 -8 -2
-9 8 1
```

```
-9 -8 -1
-9 10 2
-9 -10 -2
-10 3 3
-10 -3 -3
-10 9 1
-10 -9 -1
```

Some couples P and Q are stored more than one time because
they generate more than one real integer root.

Well, time out for me now. I'll just be watching for
other solutions.

--

Short & Sweet Math Challenges #6

*Posted by **Ex-PPC member** on **3 Mar 2003, 7:11 a.m.***

Just to light up a little this week, here's a new Short & Sweet Math Challenge (tm), for you to try on your favorite HP calc [NOT-NOT-NOT on your PC and/or Mac, that's NOT the point :-)]

The Challenge

Find a 10-digit integer number ABCDEFGHIJ where all its digits A,B,...,J must be different, such that A is divisible by 1, AB is divisible by 2, ABC is divisible by 3, and so on.

For instance, 1234567890 isn't a solution, because 1 is divisible by 1, 12 is divisible by 2, and 123 is divisible by 3, but 1234 is *not* divisible by 4.

Requirements

You may try it in any HP programmable calculator; the shortest and more elegant the program (within the selected model's capabilities, of course), the better.

Solutions and Generalization

Next week I'll provide a short program for the HP-71B that solves this challenge, as well as another program to solve the more generalized challenge of finding all numbers N, from 1-digit numbers onwards, allowing for repeated digits, that have the property that their first M-digits (from the left) form a number divisible by M (for instance, 123 would be such a solution for M=3, but 1234 wouldn't for M=4).

If you would like to tackle this generalized challenge, try to write a short program which will find and optionally print out solutions, and see if you can stablish whether there are a finite or infinite number of solutions, and in the former case what's the maximum value for M, and how many solutions are there having 1,2,..., M digits.

Re: Short & Sweet Math Challenges #6

*Posted by **Patrick** on **4 Mar 2003, 2:34 p.m.**, in response to Short & Sweet Math Challenges #6, posted by Ex-PPC member on 3 Mar 2003, 7:11 a.m.*

Here is my "first" solution. I've tried to use only features common to a bunch of HP programmables (e.g., didn't use MOD function, no recall arithmetic). This works almost verbatim on my HP-12C for example, which has one of the simpler programming models.

I haven't thought about it too hard yet, but I think this program produces the smallest number having the desired properties: 1,020,005,640.

BTW, the innocent looking subtraction in step 010 is, I think, critical to this algorithm working for all ten digits. Otherwise, it works only up to 9 digits.

Change the number in step 2 to one less than the number of digits to produce. In this version, I used 9 to produce a 10 digit result. Alternatively, have the number of digits you want in the X register when the program starts and replace the first two steps by the three steps 1, -, 10^x.

001 EEX 002 9 003 ENTER 004 1 005 STO 0 006 10 007 * 008 1 009 STO + 0 010 - 011 RCL 0 012 + 013 LAST X 014 / 015 INT 016 RCL 0 017 * 018 x<=y? (can substitute x<y? here if you like) 019 GTO 006 020 RTN

Re: Short & Sweet Math Challenges #6

*Posted by **Patrick** on **4 Mar 2003, 2:35 p.m.**, in response to Re: Short & Sweet Math Challenges #6, posted by Patrick on 4 Mar 2003, 2:34 p.m.*

Apologies for how the program listing turned out... I had them all on separate lines before I posted but they all got glued together onto one line. You can make it out with line numbers 001, 002, ... etc.

Re: Short & Sweet Math Challenges #6

*Posted by **Ex-PPC member** on **4 Mar 2003, 2:44 p.m.**, in response to Re: Short & Sweet Math Challenges #6, posted by Patrick on 4 Mar 2003, 2:34 p.m.*

Thanks for your very interesting program, Patrick, and yes, your result is correct, except for the fact that the original challenge stated:

"Find a 10-digit integer number ABCDEFGHIJ ***where all its digits A,B,...,J must be different***, such that A is divisible by 1, AB is divisible by 2, ABC is divisible by 3, and so on."

i.e.: the solution must have ten digits and all of them must be unique, no repeated digits. So, each digit 0,1,2,...,9, must appear ***once and only once*** in the number.

See if you can solve it subject to that restriction :-)

Re: Short & Sweet Math Challenges #6

*Posted by **Patrick** on **4 Mar 2003, 3:26 p.m.**, in response to Re: Short & Sweet Math Challenges #6, posted by Ex-PPC member on 4 Mar 2003, 2:44 p.m.*

Oops.

I'm on it.

Re: Short & Sweet Math Challenges #6

*Posted by **Frédéric** on **5 Mar 2003, 4:29 a.m.**, in response to Re: Short & Sweet Math Challenges #6, posted by Ex-PPC member on 4 Mar 2003, 2:44 p.m.*

What we can say ....

For ABCDEFGHIJ to be divisible by 10 => J must be equal to 0

So the problem is now for ABCDEFGHI.

For ABCDE to be divisible by 5 => E must be equal to 5

Now for the reason of odd and even numbers :

A,C,G,I are in {1,3,7,9} ....and

B,D,F,H are in {2,4,6,8}

So we must have 4!*4! combinations : 576 possibilities

This is my very little contribution.......

PS : I really like these S&SMCs ... but my old brain is really tired.

Re: Short & Sweet Math Challenges #6

*Posted by **Michael F. Coyle** on **11 Mar 2003, 5:18 p.m.**, in response to Short & Sweet Math Challenges #6, posted by Ex-PPC member on 3 Mar 2003, 7:11 a.m.*

I know it's kind of late in the game, but I do have a solution to this problem, written for the 48GX. I have to leave in a few minutes so I don't have time to post the code but I will send it to anyone interested. A brief description follows.

It solves the more general problem of finding all of the 1, 2, 3, ..., 10 digit numbers with the required properties (first N digits divisible by N, no repeated digits). It starts with a list of all the valid one-digit answers (1..9). For each of these, it forms trial 2-digit numbers by tacking on each of the digits 0..9 and seeing if the resulting 2-digit number meets the criteria; if so, it is added to a list of 2-digit solutions.

This is then repeated with the 2-digit numbers to get 3-digit solutions, and so on. This is all done in a big loop, of course. (CS fans will recognize this as a "breadth-first search.")

The program takes about 25 minutes to run and produces a list of 10 lists, giving all the solutions.

The number of solutions for each size number starts at 9 for one digit and increases to 220 for 5-digit numbers and then decreases down to just one 10-digit answer. (I will not include it here in case anyone else is still working on it.) I manually verified that the 10-digit answer and its smaller prefixes met both criteria.

Hopefully this will still be of interest to someone. For me, the fun was just getting a working program.

S&SMC#6: An HP-71B solution

*Posted by **Ex-PPC member** on **13 Mar 2003, 6:37 a.m.**, in response to Re: Short & Sweet Math Challenges #6, posted by Michael F. Coyle on 11 Mar 2003, 5:18 p.m.*

Michael wrote:

I know it's kind of late in the game, but I do have a solution to this problem, written for the 48GX [...] Hopefully this will still be of interest to someone. For me, the fun was just getting a working program.

Thanks for your interest in my challenge, perhaps it was a little more difficult than the usual "do this simple thing in the least possible number of programming steps", but it certainly offered an opportunity to show off one's favorite HP calc and some advanced programming techniques, right ?

As promised, here's a solution to both the original and the extended challenge, for the HP-71B. As usual, it's far from optimal, but it can be written very easily and gets the job done quickly:

This short (164 bytes) program recursively finds all numbers that have the property that the number consisting in the first N leftmost digits is divisible by N, for N ranging from 2 to 9 digits. Repeated digits are allowed for this version.

```
10 DESTROY ALL @ OPTION BASE 0 @ DELAY 0,0 @ STD
20 FOR I=1 TO 9 @ CALL TST((I),2,9) @ NEXT I @ DISP "DONE!" @ END
30 SUB TST(N,D,M) @ N=10*N @ FOR I=0 TO 9 @ X=N+I @ IF MOD(X,D) THEN 50
40 DISP D;X @ IF D#M THEN CALL TST(X,D+1,M)
50 NEXT I @ END SUB
```

It simply starts from 1-digit numbers and calls a subprogram that adds a new digit, testing all 10 possibilities for divisibility. When a suitable candidate appears, the subprogram calls itself, to add yet another digit, and so on till the number is 9 digits long and all possibilities have been tested. Brief comments:

- Line 10 simply sets up some initial conditions; you may want to change the DELAY to slow down the display, but the STD is important to keep the presentation tidy.
- Line 20 is the "main program": it just stablishes a loop to find solutions from 2 to 9 digits long, and calls the subprogram to do the actual work. That's it. You could search for numbers up to 12 digits long just by changing the 9 to 12 !
- Line 30 is the subprogram: it tests all 10 possibilities by adding a new digit to the already existing ones, and if the divisibility condition is met, it simply calls itself to add yet another digit
- Notice how the recursive call CALL TST(X,D+1,M) at line 40 depends on a condition, else the recursion would go ever deeper.

If you RUN this program, you'll get an output like this:

2 10 -> 3 102 -> 4 -> 1020 -> 5 -> 10200 -> 6 102000 -> 7 1020005 -> 8 10200056 -> 9 -> 102000564 -> 6 102006, etc

To find a 10-digit solution with no repeated digits, as the original challenged asked for, first we notice that for a 10-digit number to be divisible by 10, it must end in 0. So we just need to find 9-digits solutions, 0 excluded, then add a final 0 to all of them.

We just need a very few changes to the program above, like this 200+ byte version:

```
10 DESTROY ALL @ OPTION BASE 0 @ DELAY 0,0 @ STD
20 FOR I=1 TO 9 @ CALL TST((I),2,9) @ NEXT I @ DISP "DONE!" @ END
30 SUB TST(N,D,M) @ N=10*N @ FOR I=1 TO 9 @ X=N+I @ IF MOD(X,D) THEN 50
32 A$=STR$(X) @ FOR J=1 TO LEN(A$) @ IF POS(A$[J+1],A$[J,J]) THEN 50
34 NEXT J
40 DISP D;X @ IF D#M THEN CALL TST(X,D+1,M)
50 NEXT I @ END SUB
```

- As before, line 10 sets things up, but STD is even more mandatory, for the STR$ conversion below.
- The loop at line 30 in the subprogram begins at 1 instead of 0, as 0 must be the final, 10th digit.
- Lines 32 and 34 simply check for repeated digits, by converting the candidate number (which already meets the divisibility requirement) to a string, then using the POS function to find if any character occurs more than once. It's a simple, non-optimal idea that works.

The rest of the program is just the same. When RUN, it outputs:

2 12 -> 3 123 -> ... -> 9 381654729

and so, the one and only 10-digit solution is 3816547290.

That's all, hope you enjoyed it, and see how recursion makes an easy job of apparently complex tasks.

Short & Sweet Math Challenges #6 [LONG]

*Posted by **Valentin Albillo** on **24 May 2004, 1:28 p.m.***

Hi everybody, here we go (again):

Say you've just got that powerful, new HP calculator for your collection (say an HP33S or some HP48/49 model) and you're itching to test it on some interesting problem. Here's one of my own devising for you to try. Apart from being interesting in its own right (IMHO), it might actually be useful ! Consider for instance its use by someone teaching math to prepare interesting problems for his/her students (i.e.: K = 7.041776 anyone ? :-) Read on ...

Before we begin ...

We all know Pi is a *transcendental* number. By definition, this means Pi *cannot* be the root of any polynomial equation with *integer* coefficients, whatever its degree (this is also true of all other transcendental numbers such as e, sin(1), log(2), etc). In particular, Pi cannot be a root of a cubic equation with integer coefficients, such as

```
        A*x^3 + B*x^2 + C*x + D = 0
```

where A,B,C,D are all of them integer. However, this doesn't preclude the fact that such an equation might have a root which is as close an approximation to Pi as desired for sufficiently large A,B,C,D. And thus, here is

The challenge:

"Write a program for your preferred HP calculator to find the integer coefficients A,B,C,D of the cubic equation:

```
        A*x^3 + B*x^2 + C*x + D = 0
```

which has a root that best approximates a given positive constant K (say, K = Pi), where the coefficients are less or equal in absolute value to a maximum given integer value, M (say, M = 9)."

Notes:

Without loss of generality, the A coefficient will always be *positive* (i.e: it goes from 1 to M, both included) while the B,C, and D coefficients can be *either positive or negative* (i.e: they go from -M to +M, both included). Your program must *input* K (say Pi), M (say 9) and a very loose initial interval for the root (say [3.1, 3.2]) and must *output* the coefficients A,B,C,D of the "best" equation, the value X of the root, and the absolute difference between the root X and the given constant K.

For instance, given the inputs K=Pi, M=20, Interval= [3.1, 3.2] it might output something like (not the best solution, of course):

```
       A= 1, B= -10, C= 19, D= 8, X= 3.14162732179, DIF= 0.00003466820
```

which means that the cubic equation x^3 - 10*x^2 + 19*x + 8 = 0 has a root x= 3.14162732179, which differs from Pi by only 0.00003466820.

You are expected to deliver a general program, that will work for any positive constant K, maximum coefficient M and given initial interval [X1,X2], but for the purposes of testing the efficiency and accuracy of your solution you should try these two cases:

- Case 1: constant K = Pi, Max. coefficient M = 9, initial interval [3.1, 3.2]
- Case 2: same but with Max. coefficient M = 20

Hints:

- Absolutely *refrain* from using a PC. You'll learn *nothing* about writing and optimizing programs for your HP calculator and will ruin the pleasure of seeing your efforts and ingenuity succeed with your little HP machine. This is intended for *real* programmers using *real* HP calcs, not the "brute-force-using-a-zillion-Ghz-PC" kind.

- Though most any HP model can be used to successfully solve this challenge, it is advisable that you use a suitably fast HP calculator, say any model using a Saturn CPU (HP33S, HP32S, HP32SII, HP42S, HP48/49, HP-71B, etc), in order to attain reasonable running times.

- Even so, you'll need to exercise your ingenuity to achieve bearable running times. A brute-force approach is doomed to failure. Consider this:

  - for M = 9, your program will have to choose among no less than *61,731 possible cubic equations*, from (1,-9,-9-,9) to (9,9,9,9)

  - for M=20, your program will have to choose among *more than 1 million possible equations*, (namely 1,378,420) from (1,-20,-20,-20) to (20,20,20,20)

In a few days, I'll give a solution for the HP-71B, which will be a 9-line program (300+ bytes), with the following running times:

- For the case constant K = Pi, Max. coefficient M = 9, initial interval [3.1, 3.2], it takes *7 min. 16 seconds* to find the solution among the 61,731 cubic equations possible, output it *and* stop.

- For the case constant K = Pi, Max. coefficient M = 20, initial interval [3.1, 3.2], it takes *3 hour 4 min.* to find the solution among the 1,378,420 cubic equations possible, output it and stop.

Now's your turn. Just put that brand new shiny HP33S, that wonderful classical HP42S, that ultrapowerful HP49, or that venerable HP-71B to the task, *plus* your ingenuity and RPN/RPL/BASIC/FORTH/Assembler programming skills of course. After all, if the good old HP-71B can do it in 3 hours using BASIC, what incredible times will you be able to achieve with much faster machines using RPN/RPL, say ? Emulators very welcome too, of course.

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Bram** on **26 May 2004, 9:06 a.m.**, in response to Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 24 May 2004, 1:28 p.m.*

I've written a program for my HP
a real one (HP-32SII)
I found a result for M=8 as well as for M=20. The latter took about an hour for each value of A, evaluating B,C&D. (think I need new batteries now...)

I must confess however, that I didn't fulfill the assignment to the letter; I haven't "rooted" the equation for each set of parameters. I know I make a principle error here, but I think the final results will turn out the same.

Am I supposed to put my answers here?

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Valentin Albillo** on **26 May 2004, 9:16 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Bram on 26 May 2004, 9:06 a.m.*

Hi Bram:

Bram posted:

"I've written a program for my HP a real one (HP-32SII) I found a result for M=8 as well as for M=20. [...] I think the final results will turn out the same. Am I supposed to put my answers here? "

Yes, the results *and* your program for the 32SII, if you don't mind. That way we'll be able to see if your computed best solutions agree with mine and fully appreciate the programming ingenuity you used for the task, I'm pretty sure many people will be delighted to have a look at it.

As for the times, I understand it took your program some 20+ hours for the M=20 case, right ?

Thanks for your interest and best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Bram** on **27 May 2004, 7:47 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 26 May 2004, 9:16 a.m.*

By now I know that my answers are right and here's how I got them.

I already announced that I kind of cheated. Considering that the root finding part will take a lot of computing time, I decided to skip it by evaluation the function for point K for every combination of ABCD within the given limits. The combination for which the function is closest to zero, the real root will most probably be closest to K. "Most probably", because it isn't necessarily so, but as we have integer coefficients and hence the function is kind of stepping through its possible values, I thought it very likely to get the right answers all the same.
During the process every ABCD evalution that is better than the best so far, is set apart.
I realized this curious thinking in my program
When you follow the computing progress, there are times that you would like to say to the machine: "hey, don't evaluate this part any further. It's no use". Those are the parts where the program could be speeded up. F.e. when for any combination of ABC the function is already to much positive for D=-20, then you may skip the entire D-range. (I may squeeze this speed-up into the program some day).
When I finally found an ABCD combination, I did compute the real root of the function using the Solve program of my HP-32SII (EQN: D+X*(C+X*(B+X*A)))(I hate brackets ;-))
Then I subtracted Pi
and it appeared I got the right answers:
1,-1,-8,4 root=3.14133611565 err= -.000252653794
6,-16,-15,19 root=3.14159104610 err=-.00000160749

Indeed, the latter took almost 24 hours to complete, but it wwas a vary nice day.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Valentin Albillo** on **27 May 2004, 9:35 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Bram on 27 May 2004, 7:47 a.m.*

Hi Bram,

Bram posted:

"By now I know that my answers are right and here's how I got them. [...] The combination for which the function is closest to zero, the real root will most probably be closest to K. "Most probably", because it isn't necessarily so, but as we have integer coefficients and hence the function is kind of stepping through its possible values, I thought it very likely to get the right answers all the same."

You're right, very good idea. This kind of shows how plausible heuristics can go a long way toward making feasible and successful an otherwise hopeless brute-search approach.

"... when for any combination of ABC the function is already to much positive for D=-20, then you may skip the entire D-range."

Right, too. Another very good heuristic. This speeds the program immensely by essentially removing the innermost loop and thus halving (or better) processing time.

"Then I subtracted Pi and it appeared I got the right answers:"

You certainly did.

"Indeed, the latter took almost 24 hours to complete, but it was a vary nice day. "

I'm glad you took some enjoyment from my 'challenge', thanks for submitting your clever program to this forum, for using up so much juice from your batteries, and last but not least, for your interest in my humble ramblings.

Best regards from V.

Re: Short & Sweet Math Challenges #6 [(not) LONG]

*Posted by **Bram** on **30 May 2004, 3:56 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 27 May 2004, 9:35 a.m.*

I've speeded up the program. I thought it wouldm't make a big difference, but it surely does. Now I get (the same) answers in only 3 hours in stead of almost 24. The new program is with 3 lines more code significantly faster. You'll find the extension in the D-labeled part.

Re: Short & Sweet Math Challenges #6 [(not) LONG]

*Posted by **Valentin Albillo** on **31 May 2004, 6:11 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [(not) LONG], posted by Bram on 30 May 2004, 3:56 a.m.*

Sorry, Bram, but the link to your new version doesn't work.

Best regards from V.

Re: Short & Sweet Math Challenges #6 [(not) LONG]

*Posted by **Bram** on **31 May 2004, 1:29 p.m.**, in response to Re: Short & Sweet Math Challenges #6 [(not) LONG], posted by Valentin Albillo on 31 May 2004, 6:11 a.m.*

I am very very much ashamed. Please retry

Re: Short & Sweet Math Challenges #6 [(not) LONG]

*Posted by **Valentin Albillo** on **1 June 2004, 4:08 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [(not) LONG], posted by Bram on 31 May 2004, 1:29 p.m.*

No luck. This time it says "Service Unavailable".

May I suggest you simply include the whole listing in a normal post, between opening and closing "pre" codes to preserve the formatting ? It would be much simpler ...

Thanks for your efforts and

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG again]

*Posted by **Bram** on **1 June 2004, 5:02 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [(not) LONG], posted by Valentin Albillo on 1 June 2004, 4:08 a.m.*

My server at home must have tipped over; I can't reach it either (for the moment).

Anyway, the listing in directly readable form:

```
;
;       Assignment #6 for Hewlett-Packard 32SII
;       May 28, 2004
;       was placed in the MoHPC forum
;
LBL Q   ; CK=19C8 032.0
INPUT K
INPUT M
RCL M
1000
STO F
/
RCL M
+
+/-
STO E
FP
ABS
1
+
STO A
LBL A   ; CK=60AE 015.0
RCL K
3
y^x
RCL A
IP
*
STO G   ; A*x^3
RCL E
STO B
LBL B   ; CK=76A0 015.0
RCL K
x^2
RCL B
IP
*       ; B*x^2
RCL+ G
STO H   ; A*x^3+B*x^2
RCL E
STO C
LBL C   ; CK=88E4 013.5
RCL K
RCL C
IP
*       ; C*x
RCL+ H
STO I   ; A*x^3+B*x^2+C*x
RCL E
STO D
LBL D   ; CK=0550 018.0
RCL F
RCL D
```

```
IP
RCL+ I  ; A*x^3+B*x^2+C*x+D
x>y?
GTO S   ; remaining D values won't yield
ABS
x<y?
XEQ E   ; more accurate result found
ISG D
GTO D
LBL S   ; CK=5B3E 012.0
ISG C
GTO C
ISG B
GTO B
ISG A
GTO A
STOP
LBL E   ; CK=24F5 018.0
Rolldown
STO F
RCL A
STO P
RCL B
STO Q
RCL C
STO R
RCL D
STO S
RTN

total length:  123.5

;        storage registers
;        A B C D parameters
;        E counting value, typical: -M.00M
;        F distance to zero of former result
;        G H I intermediate result to speed up computation
;        K constant value for x for which the closest root should be found
;        M upper limit for all of the parameters (-M lower limit for B C D)
;        P Q R S hold parameters of best result so far
;
;        labels
;        A loop through all A values
;        B loop through all B values
;        C loop through all C values
;        D loop through all D values
;        E store results of new best result
;        Q start of program
;        S to skip irrelevant computations
```

Re: Short & Sweet Math Challenges #6 [LONG again]

Posted by ***Valentin Albillo*** *on* ***1 June 2004, 5:11 a.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG again], posted by Bram on 1 June 2004, 5:02 a.m.*

Hi, Bram:

Much better, thanks for your considerable efforts to create and share this program, which is a fine example of problem solving using a little HP32-class machine and a lot of user's ingenuity.

Let's hope you'll find my future challenges equally interesting and so you'll be tempted to contribute as well.

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Rodger Rosenbaum** on **26 May 2004, 11:12 p.m.**, in response to Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 24 May 2004, 1:28 p.m.*

Ok, here's my go at it.
It's an HP71 program, 9 lines, 344 bytes.
The print statement alone takes up 72 bytes.
I didn't use an input statement to input K and M, but just inserted them in the program
I also left in some statements to compute the time and error as it goes along.
It could have been made shorter by leaving those out.
For K=PI and M=9, it takes 2 min, 44 sec to find A=1, B=-1, C=-8, D=4, err=-.00025653792
For K=PI and M=20, it takes 26 min, 52 secs to find A=6, B=-16, C=-15, D=19, err=-.00000160749

```
10 K=PI @ M=9 @ E=MAXREAL @ T9=TIME
20 FOR P=1 TO M @ T0=P*K
30 FOR Q=-M TO M @ T1=(T0+Q)*K
40 FOR R=-M TO M @ T2=(T1+R)*K
50 IF ABS(T2)>M+.5 THEN 70
60 T=ABS(FLOOR(T2+.5)-T2) @ IF T<E THEN E=T @ A=P @ B=Q @ C=R @
D=FLOOR(.5-T2) @ PRINT E
70 NEXT R @ NEXT Q @ NEXT P @ S=(3*A*K+2*B)*K+C @ X=K @ Z=MAXREAL
80 Y=((A*X+B)*X+C)*X+D @ IF Y=Z THEN 90 ELSE Z=Y @ X=X-Y/S @ GOTO 80
90 PRINT "A=";A;"B=";B;"C=";C;"D=";D;"X=";X;"DIF=";X-K;"TIME=";TIME-T9;"SECS"
```

The program even works for M=30, but the simple little root finder in line 80
gets stuck in an infinite loop because the computed value of Y is dominated by round-off error
Replacing 'IF Y=Z THEN 90' with 'IF Y<1E-9 THEN 90' gets out of the infinite loop
but returns a less accurate root

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Valentin Albillo*** *on* ***27 May 2004, 6:54 a.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Rodger Rosenbaum on 26 May 2004, 11:12 p.m.*

Hi, Rodger:

Rodger posted:

"Ok, here's my go at it. It's an HP71 program, 9 lines, 344 bytes. [...]For K=PI and M=9, it takes 2 min, 44 sec to find A=1, B=-1, C=-8, D=4, err=-.00025653792 --- For K=PI and M=20, it takes 26 min, 52 secs to find A=6, B=-16, C=-15, D=19, err=-.00000160749"

Congratulations ! Your solutions are absolutely *correct* and your program is as *short and fast* as it gets, very good work indeed !

Just for the record, the solutions are:

M = 9

$$x^3 - x^2 - 8x + 4 = 0$$

$x = \underline{3.141}33611565$ (0.00025653794)

M = 20

$$6x^3 - 16x^2 - 15x + 19 = 0$$

$x = \underline{3.14159}104610$ (0.00000160749)

Just in case it might interest you, here is an assortment of results for larger values of M:

M = 49

$$19x^3 - 47x^2 - 30x - 31 = 0$$

$x = \underline{3.141592}35658$ (2.97E-07)

M = 99

$$33x^3 - 92x^2 - 65x + 89 = 0$$

$x = \underline{3.1415926}4441$ (9.18E-09)

M = 199

$$37x^3 - 114x^2 - 36x + 91 = 0$$

$x = \underline{3.141592653}83$ (2.39E-10)

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Dave Shaffer*** *on* ***27 May 2004, 11:33 a.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 27 May 2004, 6:54 a.m.*

Valentin,

Have you enough results (or can you generate them easily) to indicate whether the results get monotonically better (i.e. more and more accurate) with larger and larger values of M?

Actually, the answer must be NO, since for M=199, your biggest coefficient is only 114, so this must also have been the best solution for M = 115 . Similarly, for your other large-M results, the answers do not bump up to the limit set by M.

To what extent are these solutions local minima? i.e. do some of these solutions repeat for other values of M? I would think so, again based on your M = 199 solution. If I understand the challenge correctly, this solution must be the best FOR ALL VALUES OF M FROM 115 to 199. Otherwise, your answer would have a larger coefficient than 114 in it.

How far beyond 199 do you have to go until there is the next better solution?!?!

[*EDITED*] Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Valentin Albillo** on **27 May 2004, 1:28 p.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Dave Shaffer on 27 May 2004, 11:33 a.m.*

Hi, Dave:

Dave posted:

"Actually, the answer must be NO, since for M=199, your biggest coefficient is only 114, so this must also have been the best solution for M = 115."

That's correct ...

"To what extent are these solutions local minima? i.e. do some of these solutions repeat for other values of M? I would think so, again based on your M = 199 solution. If I understand the challenge correctly, this solution must be the best FOR ALL VALUES OF M FROM 115 to 199. Otherwise, your answer would have a larger coefficient than 114 in it."

Again correct. But please, don't shout ! :-)

"How far beyond 199 do you have to go until there is the next better solution?!?!"

I didn't test it extensively, but the solution for M=114 is good until M=255 (inclusive), at least. Also, a run with M=360 gives

```
   ---------------------------------------------------------
      M    a    b    c    d            x      Diff=ABS(Pi-x)
    ---------------------------------------------------------
    360  132 -319 -229 -225    3.14159265348  0.00000000011
```

which clearly is also the best solution for M=319 to M=360. The solution cubic is thus:

$$132x^3 - 319x^2 - 229x - 225 = 0$$

which has a root

$$x = \underline{3.14159265}34831+$$

which agrees with Pi to almost 11 digits. Here's some extra data for you to ponder about, this time for *e*:

```
    For K = e = 2.71828182846+
    ---------------------------------------------------------
      M    a    b    c    d            x      Diff=ABS(e-x)
    ---------------------------------------------------------
      9    3   -5   -6   -7    2.71823262630  0.00004920216
     49    6   -4  -25  -23    2.71828239152  0.00000056306
     99   29  -63  -64   57    2.71828183050  0.00000000204
    255   75 -179  -15 -143    2.71828182831  0.00000000015
```

The last line features a solution valid from M = 179 to at least 255, and the resulting equation:

$$75x^3 - 179x^2 - 15x - 143 = 0$$

has a root

$$x = \underline{2.718281828}31+$$

which is extremely close to *e*, despite the simple 3rd degree equation with integer coefficients. Either using greater degrees (say 5th-degree equations, aka *quintics*) and/or greater values for the maximum coefficients, you'll be able to achieve roots arbitrarily close to any given constant. For instance, the simple-looking equation:

$$4x^3 - 26x^2 - 11x - 30 = 0$$

has a root

$$x = 7{,}041776+ .$$

Does it ring some bells ? :-) It would make a very fine, *patriotic* assignment for any US students to solve, right ? It could also be used to construct some puzzle-like problem where after telling some story a number of conditions are given resulting in this equation, then its real root is precisely that most famous date.

Thanks for your thoughtful insights and your interest, and

Best regards from V.

Edited: 28 May 2004, 5:26 a.m.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Olivier De Smet** on **29 May 2004, 6:56 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Rodger Rosenbaum on 26 May 2004, 11:12 p.m.*

Hi,

Just some optimizations on the same topic:

With UTIL/1 binary (AWRITE) and an HP86B:

```
10 K=PI @ M=9 @ N=M+1 @ O=N*K @ P=N/K @ E=1.E99 @ CLEAR

20 FOR A=1 TO M @ T0=A*K @ IF ABS(T0)*K*K-N>O*(K+1) THEN 100

22 AWRITE 0,0 @ DISP A; T0

30 B0=MAX(-M, INT((-P-M)/K-T0+0.5)) @ B1=MIN(M, INT(P+M)/K-T0+0.5)) @ IF B0>B1
THEN 100

32 AWRITE 0,40 @ DISP B0; B1

40 FOR B=B0 to B1 @ T1=(T0+B)*K @ IF ABS(T1)*K-N>O THEN 90

42 AWRITE 1,0 @ DISP B; T1

50 C0=MAX(-M, INT(-P-T1+0.5)) @ C1=MIN(M, INT(P-T1+0.5)) @ IF C0>C1 THEN 90

52 AWRITE 1,40 @ DISP C0; C1

60 FOR C=C0 TO C1 @ T2=(T1+C)*K @ D=-INT(T2+0.5) @ IF ABS(D)>M THEN 80

62 AWRITE 2,0 @ DISP C; T2

70 IF ABS(T2+D)<E THEN F=A @ G=B @ H=C @ I=D @ E=ABS(T2+D) @ AWRITE 5,0 @
DISP F; G; H; I; E

80 NEXT C

90 NEXT B

100 NEXT A @ AWRITE 6,0 @ END
```

First it avoid looping on non-feasible A, B and C parameters.

Second it doesn't really solve the polynom, it can perhaps find a minimum.

Otherwise it's quite efficient (you can remove lines 22, 32, 42, 52, 62).

Bye,

Olivier

Edited: 29 May 2004, 6:57 a.m.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Valentin Albillo** on **31 May 2004, 6:10 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Olivier De Smet on 29 May 2004, 6:56 a.m.*

Thanks for your interest and program, Olivier, but not having an HP-86 at hand (and I suspect most MoHP regulars don't either) it would be nice if you could provide sample results and timing, to see that it delivers the correct results and also how it does compare with previous solutions.

Best regards from V.

[OT] Please, Valentin

*Posted by **Raul Lion** on **31 May 2004, 10:01 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 31 May 2004, 6:10 a.m.*

Would you be so kind as to mail me?
Drop the obvious in my account: pitquim@yahoo.es.quita.esto

Thanks in advance

Raul L

Edited: 31 May 2004, 10:02 a.m.

Certainly [No text]

*Posted by **Valentin Albillo** on **31 May 2004, 11:53 a.m.**, in response to [OT] Please, Valentin, posted by Raul Lion on 31 May 2004, 10:01 a.m.*

Best regards from V.

Pedro Perez tiene correo ;-) (nt)

*Posted by **Raul Lion** on **31 May 2004, 3:04 p.m.**, in response to Certainly [No text], posted by Valentin Albillo on 31 May 2004, 11:53 a.m.*

Gracias

[OT] No, I've checked and there's none ! :-(

*Posted by **Valentin Albillo** on **1 June 2004, 4:29 a.m.**, in response to Pedro Perez tiene correo ;-) (nt), posted by Raul Lion on 31 May 2004, 3:04 p.m.*

Hi Raul,

No new mail in the account I told you. Did the message you sent get returned back to you with an "Undelivered" error ?

Please try again. I've tested the address by sending e-mails from other accounts and it does work. If you still experience problems let me know and I'll issue a different address.

Best regards from V.

Re: [OT]I'll try again

*Posted by **Raul Lion** on **1 June 2004, 1:36 p.m.**, in response to [OT] No, I've checked and there's none ! :-(, posted by Valentin Albillo on 1 June 2004, 4:29 a.m.*

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Olivier De Smet** on **1 June 2004, 2:05 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 31 May 2004, 6:10 a.m.*

Hi,

Sorry for the delay:

For the HP86B it take about 50 sec for M=20, and 29700sec for M=200 with a better optimized program.

Anyway here is a version for an HP49G+ with all the optimizations:

```
%%HP: T(3)A(R)F(.);
\<< \pi SWAP MAXR OVER 1. + DUP 5. PICK DUP2 * UNROT / DUP 6. PICK + 7. PICK
/ { } \-> K M E N O P Q F
  \<< CLLCD K 1. M
    FOR A A 1. DISP
      IF DUP ABS K SQ * N - O K 1. + * \<=
      THEN DUP M NEG OVER Q + NEG 0. RND MAX DUP UNROT + K * SWAP M Q 4. PICK
- 0. RND MIN
        IF DUP2 \<=
        THEN
          FOR B
            IF DUP ABS K * N - O \<=
            THEN DUP M NEG OVER P + NEG 0. RND MAX DUP UNROT + K * SWAP M P
4. PICK - 0. RND MIN
              IF DUP2 \<=
              THEN
                FOR C DUP 0. RND
                  IF DUP ABS M \<=
                  THEN
                    IF DUP2 - ABS E <
                    THEN DUP2 - ABS DUP 6. DISP 'E' STO DUP NEG C B A 4. \-
>LIST REVLIST DUP 'F' STO 7. DISP
                    END
                  END DROP K +
                NEXT
              ELSE DROP2
              END DROP
            END K +
          NEXT
        ELSE DROP2
        END DROP
      END K +
    NEXT DROP E F OBJ\-> \->ARRY DUP 'S' STO
    \<< S X PEVAL
    \>> 'X' K ROOT K
  \>>
\>>
```

It take 5sec for M=9 and 40sec for M=20. Sorry I didn't test other M.

Olivier

P.S. all the results are ok with yours.

Edited: 1 June 2004, 2:08 a.m.

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Valentin Albillo*** *on* ***1 June 2004, 4:17 a.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Olivier De Smet on 1 June 2004, 2:05 a.m.*

Thank you very much, Olivier ! Magnificent effort !!

I see you master a jaw-dropping range of machines, from such an old timer as an HP-86 to the very latest HP49G+. Amazing indeed !

I did write a lot of commercial engineering programs for the HP86 and HP87 machines back in their time. Most of them were written in their built-in dialect of BASIC, but laced with a number of binary programs I wrote myself, thanks to the Assembler ROM, lots of documentation and the wonderful fact that you could have as many as 5 binary programs loaded at a time versus just one in the former HP85 models.

It was great fun when your new keywords did work ok. I specially remember a big binary I wrote which accomplished all kinds of matrix operations (at assembler speeds) but with very large, sparse matrices represented in a special format as strings ! :-) Regrettably, all those materials are "missing in action" right now. Those were the days ...

Hope to see you contributing in future S&SM Challenges I may issue, and

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Olivier De Smet*** *on* ***1 June 2004, 4:53 a.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 1 June 2004, 4:17 a.m.*

Hi,

Btw I search actually as much as possible information on the HP8x series (processor, memory controller, ...).

I have an HP85A, an HP86B some 93xx HD and a 9121 floppy. To use the 93xx I need an EMS rom, but can't find one. So I started to dis-assemble and re-assemble roms to allow them to be loaded as binary. I think it's feasible but I need some more information on the 8x series.

For example after searching the web I still have some questions:

- What does SAD and PAD opcodes exactly ? (octal 232 and 237)

- Is there an opcode for octal 326 and 336 ?

- Is there interrupts in the capricorn processor ?

- How the rom and ram paging works in an HP86 ?

- I partially understood the structure of a binary program, but I miss information for the rom structure ...

Thanks in advance if you could help me ,

Olivier

Edited: 1 June 2004, 4:54 a.m.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Eamonn** on **5 June 2004, 3:57 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Olivier De Smet on 29 May 2004, 6:56 a.m.*

To recap, the challenge is to:

> Quote:
>
> ---
>
> "Write a program for your preferred HP calculator to find the integer coefficients A,B,C,D of the cubic equation:
>
> A*x^3 + B*x^2 + C*x + D = 0
>
> which has a root that best approximates a given positive constant K (say, K = Pi), where the coefficients are less or equal in absolute value to a maximum given integer value, M (say, M = 9)."
>
> ---

I was looking at the programs offered as solutions for this challenge. While the solutions are certainly clever and very fast, they don't always find the correct polynomial. I don't have all the calculators for which the basic language solutions are provided (HP-71, HP-85, etc.), but I have tested these programs on my Sharp EL-5500III.

The cases originally chosen to test the solutions are:

> Quote:
>
> ---
>
> Case #1: constant K = Pi, Max. coefficient M = 9, initial interval [3.1, 3.2] Case #2: same but with Max. coefficient M = 20
>
> ---

For K = pi, the solutions presented do seem to return the optimal coefficients. However there are large number of values of K for which the programs do not offer the correct solution.

For example, here are two polynomials for M = 9:

f1(x) = x^3 - 5x^2 + 5x -2, which has a root at **x = 3.831177207...**

f2(x) = 3x^3 - 9x^2 -8x -6, which has a root at **x = 3.832075690...**

An exhaustive search shows that there is no other polynomial for the M = 9 case that has a root between these two roots. The midpoint of the roots of these two polynomials is **x = 3.831626449...**

Now, a correct solution to the challenge would find the coefficients for f1(x) if K is in the range **3.831177207... <= K <= 3.831626449...** and the coefficients for f2(x) if K is in the range **3.831626449... < K <= 3.832075690...**.

However the programs offered as solutions to the challenge find f1(x) for K in the range **3.831177207... <= K <= 3.83192945...** and f2(x) for K in the range **3.83192946... <= K <= 3.832075690...**.

In other words, the incorrect solution is returned for more than 33% of the values of K in the range **3.831177207... <= K <= 3.832075690...**.

The reason for this is that the programs are trying to find function f(x) that has the minimum value for f(K). This does not necessarily mean that the root for f(x) is closest to K. In this case, the value of K at which f1(K) == f2(K) is not **K = 3.831626449...** (the midpoint of the two roots), but instead is **K = 3.83192945...**

I haven't tried the HP-32s solution or the HP-48 solution, but they seem to search for the polynomial in the same way to the Basic language solutions and thus have the same problems. It would therefore appear that the challenge still remains unsolved and that a different approach to an efficient solution is required.

Note that in the original post, Valentin says:

> Quote:
>
> ---
>
> Your program must input K (say Pi), M (say 9) and a very loose initial interval for the root (say [3.1, 3.2]) and must output the coefficients A,B,C,D of the "best" equation, the value X of the root, and the absolute difference between the root X and the given constant K.
>
> ---

Since none of the solutions ask for a loose interval for the root, perhaps that is a clue to obtaining an efficient solution.

Edited: 5 June 2004, 4:00 a.m.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **hugh steers** on **6 June 2004, 3:47 p.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Eamonn on 5 June 2004, 3:57 a.m.*

thats most interesting indeed. ive been trying to come up with another way for a submission, but this point throws a spanner there too.

using the same search process, i was going to select vmin, so that whenever f(k) <= vmin, i would run a quick newton solve for f(x)=0 and compare |x-k| minimising this error. the idea was to have vmin small, so that the newton step is quick (eg one iteration). this would give the right answer...

BUT as you've pointed out f(k) could be large and yet |x-k| small with f(x)=0.

so, given its cubic and the M bound, is there is tidy upper bound for vmin that makes this work. naturally, vmin still needs be to quite small otherwise the newton process kicks in on a lot of search cases.

the other question ive been mulling, is whether there's better way altogether than the search process. i had decided to wait for valentin's answer (its less bother if you dont bother).

the only idea i came up with to accelerate the existing search idea is to note that the sign of the coeficients must change at least once when k>0 (eg for pi). this might help eliminate searches when m is larger.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Eamonn** on **8 June 2004, 2:43 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by hugh steers on 6 June 2004, 3:47 p.m.*

Hi Hugh,

I think you have some good ideas there. One way to get around the problem of large values for f(K), even when f(x) has a root close to K, may be to calculate v = |f(K)/f'(K)| and compare that to the most recent v_min.

There are not a whole lot of extra computations involved, since f'(K) is straightforward to compute (f'(x) = 3ax^2 + 2bx + c). Dividing f(K) by f'(K) should have the effect of 'normalizing' v, making it possible to do the comparison. If f'(x) is a lot less than one, then letting v = f(K) may be fine, avoiding errors that may arise from dividing by a small number.

I'm guessing that it would be necessary to compare |f(K)/f'(K)| to some multiple (2, 3 or more?) of the most recently calculated v_min to ensure that some potential solutions are not eliminated.

It also seems necessary to invoke a root finder each time a suitable polynomial is found to ensure that the correct solution is found. I don't see any easy way around that at the moment. Once a polynomial with a root closer to K has been found, v_min is updated to the v that is calculated for that polynomial.

The fastest solution will eliminate as many of the candidate polynomials as possible before calculating v. Olivier's program does a good job of reducing the number of candidate polynomials, allowing for fast execution times. Your observation of the requirement for a change of sign of at least one of the coefficients could also help execution times.

I probably won't have any time to implement these ideas myself this week, but I'm curious to see what you (or anyone else for that matter) may be able to come up with.

By the way, thanks to Valentin for formulating yet another thought provoking Short & Sweet Math Challenge.

Eamonn.

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Valentin Albillo*** *on* ***9 June 2004, 2:23 p.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Eamonn on 8 June 2004, 2:43 a.m.*

Hi, Eamonn & Hugh:

Eamon posted:

"By the way, thanks to Valentin for formulating yet another thought provoking Short & Sweet Math Challenge."

Thank you very much, I'm truly glad someone appreciates my humble challenges. Of course it would be great if more people would contribute their ideas, but I guess not so many people as I would desire are keen with things mathematic. Me, it's my #1 lifelong hobby since I can remember.

As for the challenge and the posted solutions you're right, clever and fast as they are they actually minimize the wrong quantity so they'll fail in a large number of instances, as you very cleverly notice. However, the solutions found, while not optimal, have roots very close to the desired value and also very close to the actual optimal solution. In a "production" environment (i.e.: to concoct nice problems for students to solve and such) they would be perfectly ok. But for the theoretical challenge of finding the one optimal solution they don't always deliver.

As stated in my challenge, my original solution was an HP-71B, 9-line BASIC program (314 bytes), which can solve the two test cases I gave in 7 min. 16 seconds and 3 hour 4 min. respectively. While slower than the posted solutions, mine does actually work in all cases. It goes like this:

```
  1 DESTROY ALL @ DELAY 0,0 @ DIM M,D,F,X,T,P,C,B,A,K @ INPUT
"K,M,P,T=";K,M,P,T
  2 FOR A=1 TO M @ IF SGN(((A*P-M)*P-M)*P-M)=1 THEN END
  3 FOR B=-M TO M @ IF SGN(((A*P+B)*P-M)*P-M)=1 THEN 9
  4 FOR C=-M TO M @ FOR D=-M TO M @ F=SGN(((A*P+B)*P+C)*P+D) @ IF
F=SGN(((A*T+B)*T+C)*T+D) THEN 6
  5 X=FNROOT(P,T,((A*FVAR+B)*FVAR+C)*FVAR+D) @ T=ABS(K-X) @ PRINT A;B;C;D;X;T
@ P=K-T @ T=K+T @ GOTO 7
  6 IF F=1 THEN 8
  7 NEXT D
  8 NEXT C @ NEXT B
  9 NEXT A
```

As you can see, it's just a straightforward search with several (four) heuristics added for good measure to speed the search to the point where it is manageable. It does compute the roots of the candidate polynomials, so avoiding the pitfall you demonstrated. For your example, these are the results:

```
RUN -> K,M,P,T=

       K     M  P    T                A  B  C  D        X               Err
---------------------------------------------------------------------------
   3.831626,9,3.8,3.9  [INTRO] ->  1 -6  7  5   3.83424318431   .00261718431
                               ->  1 -5  5 -2   3.83117720720   .00044879280

                                           (305 seconds)

   3.831627,9,3.8,3.9  [INTRO] ->  1 -6  7  5   3.83424318431   .00261618431
                               ->  1 -5  5 -2   3.83117720720   .00044979280
                               ->  3 -9 -8 -6   3.83207569042   .00044869042

                                           (306 seconds)
```

which are correct. P and T are the lower and upper limits of the "very loose" initial interval. The program is very simple and can be optimized further but that wasn't the intention of my challenge but to see your ideas and make you have a good time pondering it all.

By the way, Eamonn, congratulations for owning (and obviously liking) such a superb machine as your Sharp EL-5500III. I'm also a Sharp collector and like them a lot. They clearly meet and surpass many HP models past and present, in quality, features and usability as well. May I recommend you also get hold of a Sharp PC-1475, which has all the functionality of your EL-5500III plus a large two-line display and 20-decimal, double precision. Also, the Sharp PC-1350, with its 4-line, fully graphic display is a wonder to behold and use.

Thanks again for your incredibly keen observations and interest and

Best regards from V.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **bill platt** on **9 June 2004, 3:29 p.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 9 June 2004, 2:23 p.m.*

Hi Valentin!

> Quote:
>
> ---
>
> Of course it would be great if more people would contribute their ideas, but I guess not so many people as I would desire are keen with things mathematic.
>
> ---

Actually, I rather suspect that there are many out there like me who appreciate these challenges, and read them, along with the remarkable answers, with great interest---we merely do not have the skills....but we enjoy watching genius unfold, and hope to glean at least a few grains from the experience.

Of course I hope that someday I will manage to have a good attempt that will be worthy of posting!

Best regards,

Bill

Edited: 9 June 2004, 3:30 p.m.

Re: Short & Sweet Math Challenges #6 [LONG]

*Posted by **Bill Smith** on **10 June 2004, 11:44 a.m.**, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Valentin Albillo on 9 June 2004, 2:23 p.m.*

I too, want to add that these challenges are great entertainment. Usually I catch up on the forum while finishing my morning coffee at work. But unfortunately work always presses, and I don't presently have home internet service to pursue the challenge on my own time.

Someday I hope to have a chance to put my solution efforts here, so keep the challenges coming! Thanks, and best regards,

bs

---

*Posted by **Valentin Albillo** on **10 June 2004, 12:30 p.m.**, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Bill Smith on 10 June 2004, 11:44 a.m.*

I do really appreciate very much your kind feedback, as it lets me know that my humble efforts do really reach like-minded people, even if they don't choose to post a reply to the particular challenge.

I always aim to be didactic (though at times I may sound 'bombastic' instead) so "readers" are as precious to me as "contributors", to be sure.

More are on their way, stay tuned ! :-)

Best regards from V.

Re: I am in same group as the bianary Bills.

Posted by ***Ron Ross*** *on* ***10 June 2004, 2:03 p.m.****, in response to* Thank you very much, Bill & Bill :-)*, posted by Valentin Albillo on 10 June 2004, 12:30 p.m.*

I too, do not always have (or make) time to tackle your challenges, but they are always informative and I do sometimes learn (admitidly I don't always learn, but thats because many of the solutions are above me).

My name on the list, too, please!

Posted by ***Vieira, Luiz C. (Brazil)*** *on* ***10 June 2004, 5:32 p.m.****, in response to* Thank you very much, Bill & Bill :-)*, posted by Valentin Albillo on 10 June 2004, 12:30 p.m.*

Hi Valentin, guys;

I feel I'm a bit of a "math Voyeur", the respectfull kind, of course... >8^O

Please, Valentin, don't take us, at the "silence tribune", as a reference; instead, consider the brave ones that go further and answer your chalenges. When I figure a chance to add some positive comments or possible solutions, I "break the silence" and dare (still in the good sense) posting. Otherwise, I keep my "math Voyerism" in practice...

Best regards from Brazil.

Luiz

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by Veli-Pekka Nousiainen *on* ***10 June 2004, 5:12 p.m.****, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Valentin Albillo on 9 June 2004, 2:23 p.m.*

```
Nice program to a nifty problem!
Has anybody tried to convert that for the 49 series?

Do you, Valentin, have an example with 10 different values as a result?
[VPN]
```

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Valentin Albillo*** *on* ***11 June 2004, 5:37 a.m.****, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Veli-Pekka Nousiainen on 10 June 2004, 5:12 p.m.*

Hi, VPN:

VPN posted:

"Do you, Valentin, have an example with 10 different values as a result?"

Excuse me but I don't actually understand your question. Could you elaborate, please ?

P.D.: No disrespect intended but as a Systems Engineer I actually find it funny your VPN moniker (Virtual Private Network] :-)

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by Veli-Pekka Nousiainen *on* ***11 June 2004, 9:14 a.m.****, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Valentin Albillo on 11 June 2004, 5:37 a.m.*

```
Never mind!
A) I'm doing the program myself
B) with M=20 one gets 10 "answers"
[Virtual Polish Notation]
```

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Ángel Martin*** *on* ***11 June 2004, 11:12 a.m.****, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Veli-Pekka Nousiainen on 11 June 2004, 9:14 a.m.*

I thought last time I saw it your moniker stood for "vertical"(not virtual) polish notation...

identity crisis, anyone?

Re: Short & Sweet Math Challenges #6 [LONG]

Posted by ***Veli-Pekka Nousiainen*** *on* ***11 June 2004, 12:03 p.m.****, in response to Re: Short & Sweet Math Challenges #6 [LONG], posted by Ángel Martin on 11 June 2004, 11:12 a.m.*

My true identity revealed:
[Visual Processing Node]
(-;

[VPN]: Short & Sweet Math Challenges #6 [49-series]

Posted by Veli-Pekka Nousiainen *on* ***19 June 2004, 9:26 a.m.****, in response to* Re: Short & Sweet Math Challenges #6 [LONG]*, posted by Veli-Pekka Nousiainen on 11 June 2004, 9:14 a.m.*

```
%%HP: T(3)A(R)F(.); \<< "SOLVE" { { "K:" "K: guess" 0. } { "P:" "P: min" 0. } { "T:" "T:
max" 0. } { "M:" "M: \177loop" 0. } } { } { } { } IF INFORM THEN { K P T M } \->TAG DUP
LIST\-> 0 DUPDUP TIME \-> K P T M a b I F t \<< PUSH STD { -1. -55. -72. } SF { -56. -105.
} CF CLLCD DATE TIME TSTR 'R.L' DUP2 STO ROT "0" \->TAG 1. \->LIST STO+ 15. 22.
SUB DUPDUP 1. 2. SUB { # 0h # 0h } 0. DISPXY ":" { # 7h # 0h } 0. DISPXY 4. 5. SUB { #
Ah # 0h } 0. DISPXY ":" { # 11h # 0h } 0. DISPXY 7. 8. SUB { # 14h # 0h } 0. DISPXY
"CURR." { # 0h # 7h } 0. DISPXY "-----" { # 0h # Ch } 0. DISPXY "A:" { # 0h # 11h } 0.
DISPXY "B:" { # 0h # 18h } 0. DISPXY "C:" { # 0h # 1Fh } 0. DISPXY "D:" { # 0h # 26h } 0.
DISPXY 1. M FOR A " " A DUP 'a' STO + SREV TAIL 1. 3. SUB SREV { # 8h # 11h } 0.
DISPXY IF M P A OVER * PICK3 - OVER * PICK3 - * OVER - SIGN 1. == THEN 'A' STO+
ELSE NEG M FOR B " " B DUP 'b' STO + SREV TAIL 1. 3. SUB SREV { # 8h # 18h } 0.
DISPXY IF M P A OVER * B + OVER * PICK3 - * SWAP - SIGN 1. == THEN M 1. + 'B'
STO ELSE M NEG M FOR C " " C + SREV TAIL 1. 3. SUB SREV { # 8h # 1Fh } 0. DISPXY
M NEG M FOR D " " D + SREV TAIL 1. 3. SUB SREV { # 8h # 26h } 0. DISPXY P A OVER
* B + OVER * C + * D + SIGN 'F' STO IF T A OVER * B + OVER * C + * D + SIGN F ==
THEN IF F 1. == THEN M 1. + 'D' STO END ELSE 9999 .1 BEEP K '((A*X+B)*X+C)*X+D'
'X' P T 2. \->LIST ROOT - ABS 'T' STO 'I' INCR 1 - 3 MOD 19 * 9 + R\->B TIME t HMS- IF
DUP 0. < THEN 24. + END \-> y h \<< 'R.L' { A B C D X T h } DUP EVAL 7. \->LIST SWAP
\->TAG I \->STR \->TAG 1. \->LIST STO+ "A:" " " A + SREV TAIL 1. 3. SUB SREV + " B:"
" " B + SREV TAIL 1. 3. SUB SREV + + " C:" " " C + SREV TAIL 1. 3. SUB SREV + + " D:"
" " D + SREV TAIL 1. 3. SUB SREV + + { # 28h } y # 9h - + 0. DISPXY " " I + SREV 1. 2.
SUB SREV " X:" + X + { # 1Ch } y # 3h - + 0. DISPXY DATE h TSTR 15. 22. SUB DUPDUP
1. 2. SUB { # 68h } y + 0. DISPXY ":" { # 6Fh } y + 0. DISPXY 4. 5. SUB { # 72h } y + 0.
DISPXY ":" { # 79h } y + 0. DISPXY 7. 8. SUB { # 7Ch } y + 0. DISPXY "T:0" T + { # 28h } y
# 3h + + 0. DISPXY \>> K T DUP2 - 'P' STO + 'T' STO END NEXT NEXT END NEXT END
NEXT POP 999. .1 BEEP "=====" { # 0h # 2Ch } 0. DISPXY DATE TIME TSTR 'R.L' OVER
STO+ 'R.L' { a b } DUP EVAL 2. \->LIST SWAP \->TAG "\oo" \->TAG 1. \->LIST STO+ 15.
22. SUB DUPDUP 1. 2. SUB { # 0h # 32h } 0. DISPXY ":" { # 7h # 32h } 0. DISPXY 4. 5.
SUB { # Ah # 32h } 0. DISPXY ":" { # 11h # 32h } 0. DISPXY 7. 8. SUB { # 14h # 32h } 0.
DISPXY 0. WAIT DROP \>> END \>>
```

Short & Sweet Mathematical Challenge #7

Posted by ***Valentin Albillo*** *on* ***5 July 2004, 9:27 a.m.***

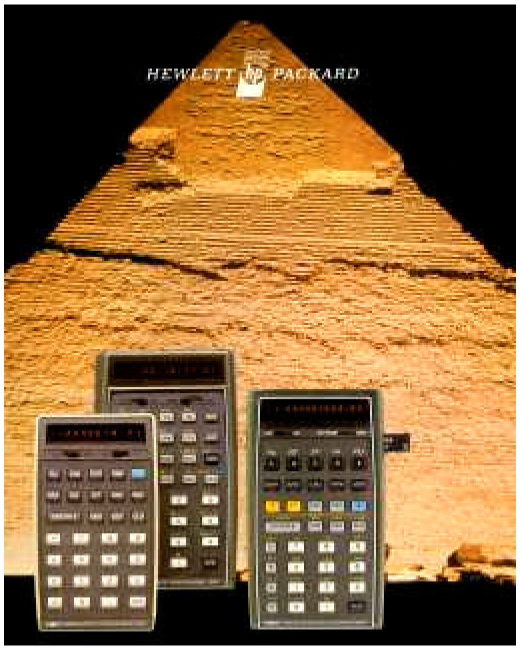
Hi all, a new month begets a new S&S Math Challenge, #7 to be precise.

This time you'll have to use to the max all your math intuition, skills, a lot of sleuthing, and a respectable amount of programming ingenuity though, as always, the required program won't be neither too difficult nor too long.

I can assure you that the answer to this particular challenge will take you *absolutely by surprise*, you'll be *enthralled* by the discovery process, then absolutely *delighted* with the unexpected outcome. Matter of fact, you might want to tackle this challenge all by yourself, refraining from reading any posts in this thread till you find the solution on your own, so as to avoid spoiling the (considerable) pleasure. This is truly one of a kind, trust me !

So, it seems this would be reward enough for any HP/Math fan, right ?

Well, no. Just in case this isn't enough for you, there's yet another interesting (if material) reward: a high-quality 1280 x 1024 image of this fabulous classic "poster", scanned by myself, depicting three Classical and Woodstock HP machines against an ancient pyramidal background, have a look at this small thumbnail:



This magnificent 1280 x 1024 image will be awarded to the first person (or persons, eventually) who comes out with a working program and correct answers to the questions raised by

The Challenge

This Challenge is structured in two parts. *First*, you must write a program to compute a given function. *Then*, you must use your program *and* intuition to answer a number of preparatory, training questions about the function, ending in a **Big Question[tm]** which will require all your sleuth abilities as well as data gathered from the answers to the training questions.

First part: The Program

Use your favorite HP calculator (most any will do) to implement the following function **h(n)**, defined for integer n greater than 0 as follows:

```
h(1)    = 1
h(3)    = 3
h(2n)   = h(n)
h(4n+1) = 2*h(2n+1)-h(n)
h(4n+3) = 3*h(2n+1)-2*h(n)
```

Your program must accept **n** as input and return **h(n)** as output, where n is an arbitrary integer greater than 0.

If you succeed, you'll be in a most favorable position to answer the following *training* questions, which will increase your chances of finding a correct answer to the ultimate last question.

Second part: The Questions

Use your program to compute **h(n)** and a little sleuthing to answer these training questions, *in order*. They are designed to led you effortlessly, step by step, to the **Big Question[tm]**. (m and n are always arbitrary integers greater than 0).

1. What's the value of h(2^n) ? ditto h(2^n + 1) ? h(2^n - 1) ? h(m * 2^n) ?
2. For which n is h(n) = 100 ?
3. For which n is h(n) even ?
4. If n is odd and m = h(n), what's the value of h(m) ?
5. What are the values of h(h(n)) ?
6. For which values of n do we get h(n) = n ?

Second part encore: The Big Question [tm]

Just what on Earth does h(n) to n ??

This is, describe the simplest relationship the result value has with the given argument. In English (or Spanish) you can state this relationship unambiguously using just ***three words***. Which ones ?

Notes:

In order to check that your program delivers correct results for h(n), you might try to check out the following random cases:

```
     n  h(n)
  ------------
   841  587
   642  261
    44   13
   821  691
   381  381
    17   17
   378  189
   226   71
    86   53
   742  413
   765  765
   514  257
```

Give it a try, I'm sure you'll enjoy it ! :-)

Best regards from V.

Re: Short & Sweet Mathematical Challenge #7

*Posted by **J-F Garnier** on **5 July 2004, 12:34 p.m.**, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

Hi Valentin,

First of all, my test program on my HP-71B. I used user-defined function and recursivity, close to the function definition:

```
10 DEF FNH(N)
20 IF N=1 THEN FNH=1 @ END
30 IF N=3 THEN FNH=3 @ END
40 IF MOD(N,2)=0 THEN FNH=FNH(N/2) @ END
50 IF MOD(N,4)=1 THEN FNH=2*FNH((N-1)/2+1)-FNH((N-1)/4) @ END
60 FNH=3*FNH((N-3)/2+1)-2*FNH((N-3)/4)
70 END DEF
```

Maybe the implementation on a RPN, limited-stack machine would be more tricky. It could be a challenge by itself.

I succeeded to find the relation between n and h(n), but I don't want to spoil the game right now. I will just say that fnh(168788912877) is 196856925369 (quite hard to check with the recursive method) and that the HP-71B Math ROM includes a convenient function to check this.

It was really a pleasure to spend some time on your challenge.

J-F

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Bram** on **5 July 2004, 1:32 p.m.**, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

Hi Valentin,

I already found the relation between n and h(n) and my phrase would be E.A.R.
I'll explain this later, because I first have to write my program. I manually compiled a table for n=1..20 and I discovered the relation fairly quickly.
Knowing the relation I could easily write a non-recursive routine, but I will try to write a program as if I don't know anything about it yet.

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Bram** on **5 July 2004, 2:57 p.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Bram on 5 July 2004, 1:32 p.m.*

Well, no program yet. Quite a challenge for an HP-32SII without recursion, but I won't wait for it to explain the E.A.R. in my previous post.

So here's the relation I've found

DON'T READ ON IF YOU DON'T YET WANT TO KNOW

DON'T READ ON IF YOU DON'T YET WANT TO KNOW

FINAL WARNING

Extract And Reverse is how I've put it.
Take n
Convert it to binary
Extract the part from the first 1 until the last 1 (so strip leading and trailing zeros)
Reverse this string of binary digit(s)
Convert back to decimal
Et voilá

And that explains why there aren't any even results; you'll always have a 1 as the LSB, hence an odd number.
And also why 2^n+1 doesn't change; two ones separated by zeros
And also why 2^n-1 doesn't change; all ones

Thanks Valentin, for this well spent night (local time is 21:00)

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Veli-Pekka Nousiainen** on **6 July 2004, 12:02 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Bram on 5 July 2004, 2:57 p.m.*

```
I present only a HP-49G program (not perfect :)
<< -> n
   << n
      IF DUP 0 >
      THEN PUSH BIN R->B ->STR 3
OVER SIZE SUB SREV TAIL "#" SWAP
+ STR-> B->R R->I POP
      END n
   >> DROP
>>
BYTES => # 14775o 111.5
<< VPN >>
```

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Carl Nelson** on **9 July 2004, 2:48 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Veli-Pekka Nousiainen on 6 July 2004, 12:02 a.m.*

For HP-48G:

<< BIN 64 STWS R->B #0b SWAP

WHILE DUP B->R REPEAT SR LASTARG #1b AND ROT SL OR SWAP END

DROP B->R >>

I also did a solution for the HP-11C, HP-12C, and HP-42S, but I don't have the listings handy here.

--Carl

Massimo's Last Theorem

*Posted by **Massimo (Italy)** on **9 July 2004, 4:10 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Carl Nelson on 9 July 2004, 2:48 a.m.*

> Quote:
>
> ---
>
> also did a solution for the HP-11C, HP-12C, and HP-42S, but I don't have the listings handy here
>
> ---

I found a clever solution for the HP-01 but the margin on my copy of this book by Diophantus is too small...

:-)

Re: Massimo's Last Theorem

*Posted by **Ángel Martin** on **9 July 2004, 7:57 a.m.**, in response to Massimo's Last Theorem, posted by Massimo (Italy) on 9 July 2004, 4:10 a.m.*

You need to stop buying those chipo books with such small margins. Try the coffe-table edition next time !

Best, ÁM

Re: Massimo's Last Theorem

Posted by ***Carl Nelson*** *on* ***10 July 2004, 12:52 a.m.****, in response to Re: Massimo's Last Theorem, posted by Ángel Martin on 9 July 2004, 7:57 a.m.*

Wise guys, eh? :)

Here is the code for the HP-42S. It's the most compact. The 11C AND 12C are similar, except for the issue of labels (single letter/number on 11C, nonexistant on 12C, use line numbers for GTOs), and lack of MOD function on both (use 2 / frac 2 *)

```
LBL “HOFN”
0
STO 00
(ROLLDOWN)
LBL 01
X=0?
GTO 02
ENTER
ENTER
2
STO * 00
MOD
STO + 00
(ROLLDOWN)
2
/
IP
GTO 01
LBL 02
RCL 00
R/S
```

Re: Massimo's Last Theorem

Posted by ***Massimo (Italy)*** *on* ***10 July 2004, 7:43 a.m.****, in response to Re: Massimo's Last Theorem, posted by Carl Nelson on 10 July 2004, 12:52 a.m.*

> Quote:
>
> ---
>
> Wise guys, eh? :)
>
> ---

No offence taken, I hope. I just couldn't resist after reading your original post.

Massimo

Re: Short & Sweet Mathematical Challenge #7

*Posted by **RMillán** on **5 July 2004, 7:32 p.m.**, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

Hi, Valentín:

While thinking about how to program this *h* function in a 15C, I'll post a simple program for the 16C which will compute the function:

```
001 LBL A
002 0
003 x<>y
004 LBL 0
005 x!=0
006 GTO 1
007 RDN
008 RTN
009 LBL 1
010 SR
011 x<>y
012 RLC
013 x<>y
014 GTO 0
```

It works in any wordsize, but of course 64 will give the maximum numeric range.

I suppose that using the 16C for this challenge is kind of cheating. Indeed, I did not begin, as was required, by programming the function in *my favorite HP calculator (most any will do)*; among the non-RPL calculators, only the 16C allows a trivial implementation.

The challenge is now for me to find a reasonably implementation in the 15C. Also, I'm not so sure I could express the relation (in English or in Spanish) in just three words.

Regards,
Rafael Millán.

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Cameron** on **6 July 2004, 5:04 a.m.**, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

I suspect the horse has bolted (judging from the number of responses). However I'll make my contribution and await for Valentin to declare a winning entry before I look at the others.

Normally I treat these challenges as a spectator sport. They're typically most illuminating but require a prowess in mathematics that I don't posess. The word that caught my eye on this one was *integer*. I know a thing or two about (the computer representation of) integers. So I read the challenge a little more closely than usual and decided I could produce a solution.

Valentin said to use my favorite HP, so I fired up my 16C simulator (it works on the real one too). Here's the program:

```
001     LBL FN_H        ;       n
002     [0]             ;       h=0     n
003     X-Y             ;       n       h
004     LBL LOOP        ;       n       h

; shift LSB out of n into carry { carry = n & 1; n >>= 1; }

005     SR              ;       n=n/2   h
006     X-Y             ;       h       n

; shift LSB into h from carry { h = h<<1 | carry; }

007     RLC             ;       h=RLC   n
008     X-Y             ;       n       h

; Any bits left in n?

009     X!=0?
010     GTO LOOP

; cleanup--result into X

011     X-Y             ;       h       0
012     (RTN)           ; not necessary if last in memory
```

And now the answers to the questions:

> Quote:
>
> ---
>
> 1. What's the value of $h(2^n)$? ditto $h(2^n + 1)$? $h(2^n - 1)$? $h(m * 2^n)$?
>
> ---

1a: 1
1b: $2^n + 1$
1c: $2^n - 1$
1d: if m is odd, m else if m == 2^x, 1 else h(n)

> Quote:
>
> ---
>
> 2. For which n is h(n) = 100 ?

h(n) can never be 100.

Quote:

3. For which n is h(n) even ?

h(n) can never be even.

Quote:

4. If n is odd and m = h(n), what's the value of h(m) ?

n

Quote:

5. What are the values of h(h(n)) ?

n

Quote:

6. For which values of n do we get h(n) = n ?

$2^x - 1$

Before I answer the big question^{TM} I have to confess to having cheated. Valentin invited us:

Quote:

to check out the following random cases:

I looked carfully at those numbers. If indeed they were "random" and not carfully contrived, I could see the (bit) pattern that led to my conclusion about the defining property of h(n).

Now, the ultimate answer: right-justified bit reversal.

Of course that's not what Valentin was looking for. I've wracked my brains to weave the "Earth" hint into the answer but it eludes me. I am sure that someone who more clever with word-play will get it.

BTW, right-justified bit reversal is a technique that is used in permuted index generation. One applicatition that springs to mind is DFT. I'm sure Knuth wrote about it in *Sorting and Searching*.

Cameron

PS: a corollary challenge. Write a program to compute the probability that Valentin's (random) test cases were all right-justified reversals.

Re: Short & Sweet Mathematical Challenge #7

*Posted by **J-F Garnier** on **6 July 2004, 8:08 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Cameron on 6 July 2004, 5:04 a.m.*

My answer to question #6 would be 'n is a binary palindrome'.

Regarding DFT, I think that the reversal is NOT right-justified (otherwise no even index would be generated).

J-F

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Cameron** on **6 July 2004, 8:48 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by J-F Garnier on 6 July 2004, 8:08 a.m.*

> Quote:
>
> ---
>
> My answer to question #6 would be 'n is a binary palindrome'.
>
> ---

h(n) is a palindrome iff n is a palindrome.

> Quote:
>
> ---
>
> Regarding DFT, I think that the reversal is NOT right-justified (otherwise no even index would be generated).
>
> ---

D'oh! You are, of course, correct. Apologies to all for the misinformation.

Cameron

definition of binary palindrome?

*Posted by **Cameron** on **7 July 2004, 3:43 p.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by J-F Garnier on 6 July 2004, 8:08 a.m.*

It occurs to me that my notion of palindrome may be different to everyone else's. As a computer guy, I tend to think of all bits (in a given parcel) as being significant. Consider this octet:

00001001

To my mind, that is *not* a palindrome although its right-justified reversal will have the same value. This demonstrates the flaw in my thinking. Since the leading zeros are numerically insignificant, it is a palindrome (although I would still contend that the octet *holds* a palindrome).

In light of this, your answer to question 6 would appear to be correct.

Comments?

Cameron

Ahhrrrg! A typo.

*Posted by **Cameron** on **6 July 2004, 8:58 a.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Cameron on 6 July 2004, 5:04 a.m.*

> Quote:
>
> ---
>
> ```
> Quote:
> ------------------------------------------------------------------
> ---------------
>  6. For which values of n do we get h(n) = n ?
> ------------------------------------------------------------------
> ---------------
> ```
> $2^x - 1$
>
> ---

This should read:

$2^{x} {}^{+}/- 1$

Serves me right for dinking with fancy formatting.

Cameron

Question#6

*Posted by **Tizedes Csaba [Hungary]** on **6 July 2004, 10:51 a.m.**, in response to Ahhrrrg! A typo., posted by Cameron on 6 July 2004, 8:58 a.m.*

I dont want to provoke - because I havent got answer - but what about n=381 and n=765...?!

Csaba

Ps.: I am working on my solution. I dont believe in this 'binary-palindrom'... Its just pure mathematics... ;) But not evident...

Re: Question#6

*Posted by **Cameron** on **6 July 2004, 11:11 a.m.**, in response to Question#6, posted by Tizedes Csaba [Hungary] on 6 July 2004, 10:51 a.m.*

Well spotted! <Gently lays queen on her side>

It's obvious now when I reconsider it. All the odd palindromes < $2^{max} - 1$ must satisfy h(n) == n. Is there a simple expression that defines the series?

I'm off to bed. I'll check back in the morning.

Cameron

Re: Short & Sweet Mathematical Challenge #7

*Posted by **Cameron** on **7 July 2004, 4:19 p.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Cameron on 6 July 2004, 5:04 a.m.*

Obviously my answer to 5 is only true if n is odd. The series that results from h(h(n)) for even n is fascinating. However I don't know what to make of it.

Cameron

Re: Short & Sweet Mathematical Challenge #7

*Posted by **W Rice** on **7 July 2004, 8:37 p.m.**, in response to Re: Short & Sweet Mathematical Challenge #7, posted by Cameron on 7 July 2004, 4:19 p.m.*

For n even, it's the first odd number you encounter after continually dividing by 2.

No solution yet, just two equation...

*Posted by **Tizedes Csaba [Hungary]** on **8 July 2004, 7:36 a.m.**, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

It's trivial, but maybe(???) this will help somebody:

```
[[ 3  -2 ]  * [[ h(4*n+1) ]  =  [[   h(n)   ]
 [ 2  -1 ]]    [ h(4*n+3) ]]     [ h(2*n+1) ]]
```

Csaba

S&SMC #7: My Solutions And Comments [LOOOOONG]

Posted by ***Valentin Albillo*** *on* ***8 July 2004, 8:49 a.m.****, in response to Short & Sweet Mathematical Challenge #7, posted by Valentin Albillo on 5 July 2004, 9:27 a.m.*

Thanks to all of you interested in my Challenge, whether 'posters' or 'lurkers'. Questions have been answered correctly and some working programs have been produced, though some of you 'cheated' and found the answers by analyzing the test cases I gave, instead of actually implementing h(n) as defined, which was the intended exercise. Well, so much for giving test cases, now for my solution and comments:

Note: All the code herein is for the HP-71B because its English-like BASIC syntax makes it very easy to understand and vey easy to translate to RPN/RPL, while the reverse isn't exactly true. Also, even if you haven't got an HP-71B, you can try and run all the code and examples featured here by using Emu71, Jean-François Garnier's superb HP-71B emulator which you can download for free from that link.

First part: The Program

The Challenge required you to implement h(n) defined for integer n > 0 as follows:

```
h(1)    = 1
h(3)    = 3
h(2n)   = h(n)
h(4n+1) = 2*h(2n+1)-h(n)
h(4n+3) = 3*h(2n+1)-2*h(n)
```

If your HP machine does natively support *recursion*, this is very easy to implement, like this HP-71B version, which defines a user-defined function FNH(N) which computes h(n) by making recursive calls to itself:

```
10 DEF FNH(N) @ IF N=1 OR N=3 THEN FNH=N @ END
20 ON RMD(N,4)+1 GOTO 30,40,30,50
30 FNH=FNH(N/2) @ END
40 FNH=2*FNH((N+1)/2)-FNH((N-1)/4) @ END
50 FNH=3*FNH((N-1)/2)-2*FNH((N-3)/4) @ END DEF
```

Implementing h(n) in an HP calc without recursion is much more *tricky*, and usually requires dealing with stacks to keep the parameters and returns for the ever deeper recursive calls, but once you get the knack of it, it's actually fairly straightforward.

However, for this particular challenge, where implementing the function is just for the sake of using it to do the sleuthing and answer the questions, we can make do with a much simpler technique, which will allow us to *instantly* get the value of h(n) for a range of n values large enough for our purposes, and does not require recursion at all so it can be easily implemented in most any HP calculator this side of the HP-67 or so. Instead of recursion, we'll use a suitably dimensioned array (or range of registers) to keep the function values, which will be constantly reused to generate further values, substituting recursion for *direct array lookup*, like this:

```
10 DESTROY ALL @ OPTION BASE 1 @ INTEGER H(1000)
20 FOR N=1 TO 1000 @ IF N=1 OR N=3 THEN H(N)=N @ GOTO 70
30 ON RMD(N,4)+1 GOTO 40,50,40,60
40 H(N)=H(N/2) @ GOTO 70
50 H(N)=2*H((N+1)/2)-H((N-1)/4) @ GOTO 70
60 H(N)=3*H((N-1)/2)-2*H((N-3)/4) @ GOTO 70
70 NEXT N @ DISP "Ready: Use FNH(n) (1 <= n <= 1000)" @ END
80 DEF FNH(N)=H(N)
```

Notice just how closely does this mimic the recursive version, only using H(n) (the array) instead of FNH(n) (the recursive call). This version does pre-compute h(1) to h(1000) and stores the results in array H, to be used by the user-defined function FNH(n), just like in the recursive version. Actually, you could use just H(n) directly, instead of FNH(n), but this way both versions work alike and the following examples using them are the same. Don't forget to RUN this version for it to generate the values (RUN is not needed for the recursive defininition).

This makeshift, non-recursive version generates all 1000 values very fast, and using FNH afterwards is much, much faster; however, it does take more memory (to store the values) and is limited to the range 1-1000, while the recursive version doesn't have any such limitation. But for quickly and effortlessly implementing a recursive definition in a hurry, this array technique does deliver the goods !

Second part: The Questions

Using our newly implemented FNH, we can do some sleuthing to try and answer the questions, namely:

1. *"What's the value of h(2^n) ? ditto h(2^n + 1) ? h(2^n - 1) ? h(m * 2^n) ?"*

   For the first question, we will try this, from the command line:
   ```
       FOR N=1 TO 5 @ X=2^N @ DISP X,FNH(X) @ NEXT N
   ```
   You'll get all 1's, so it seems that h(2^n) = 1.

   Similar command lines will quickly reveal that h(2^n+1) = 2^n+1 and h(2^n-1) = 2^n-1. As for h(m*2^n), sampling a few values like this:
   ```
       FOR M=14 TO 18 @ FOR N=1 TO 3 @ X=M*2^N @ DISP M;FNH(M);X;FNH(X) @
   NEXT N @ NEXT M
   ```
   will make it crystal-clear that h(m*2^n) = h(m).

2. *"For which n is h(n) = 100 ?"*

   Sampling any random values or range of values makes it obvious that h(n) in always odd, so it can never be an even value like 100.

3. *"For which n is h(n) even ?"*

   Ditto. h(n) is always odd.

4. *"If n is odd and m = h(n), what's the value of h(m) ?"*

   Sampling a random range of odd n values, like this:
   ```
       FOR N=431 TO 451 STEP 2 @ DISP N;FNH(FNH(N)) @ NEXT N
   ```
   makes it absolutely clear that in this case h(m) = n.

5. *"What are the values of h(h(n)) ?"*

   Ditto. As we've seen in the previous question, if n is odd then h(h(n)) = n. If n is even, then h(h(n)) is n with all factors 2 casted out.

6. *"For which values of n do we get h(n) = n ?"*

   This one is tricky. As we've seen in question (1) above, if n is a power of two plus or minus one, then h(n)=n. But are there any other values apart from these ? Let's try, add these lines to the program:
   ```
       100 FOR N=100 TO 300 @ IF FNH(N)=N THEN DISP N;
       110 NEXT N @ DISP
   ```
   Now running it (using RUN 100, to avoid clearing the array in the case of the non-recursive version), produces at once:
   ```
       107  119  127  129  153  165  189  195  219  231  255  257  273
   297
   ```

where the powers of 2 plus/minus one do appear (127-129, 255-257), plus a host of other values, but though it's evident that there's some *structure* to it all, it isn't clear what it might be.

However, there are some *signs* here and there waiting for us to notice. Though the function's definition doesn't seem to have much to do with powers of 2, we've found that such arguments are indeed *special*. Exact powers of 2 are always smashed to 1, while their predecessors and successors are returned *intact*. Might it be the case that working in base-10 is actually confusing the issue ? How would it all look *if we used base-2* instead, where powers of 2 are very 'round' numbers (10,100,1000,10000, ...) ?

Let's try ! We'll show both arguments and results in base-2 and see how it all looks like. Altering line 100 above like this:

```
    100 FOR N=100 TO 300 @ IF FNH(N)=N THEN DISP N;BSTR$(N,2)
```

and running it again (using RUN 100), we get these results:

```
    107   1101011
    119   1110111
    127   1111111
    129   10000001
    153   10011001
    165   10100101
    189   10111101
    195   11000011
    219   11011011
    231   11100111
    255   11111111
    257   100000001
    273   100010001
    297   100101001
```

where it is obvious that all n being returned intact by h(n) share a common trait: **their binary representations are palindromic !** Reversing their bit-order leaves them unchanged, they are the mirror images of themselves.

Second part encore: The Big Question [tm]

"Just what on Earth does h(n) do to n ??"

Well, once we've empirically explored how h(n) works and with our intuition sharpened by our previous findings, it's only natural for us to make a bold final statement:

<u>Given an argument n, h(n) massages it a little while and then simply, in three words, **reverses its bits**.</u>

As further confirmation, this command line:

```
    FOR N=121 TO 137 STEP 2 @ DISP N,BSTR$(N,2),FNH(N),BSTR$(FNH(N),2)  @
NEXT N
```

produces:

```
     n     n (base2)    h(n)   h(n) (base2)
   ----------------------------------------
    121     1111001      79      1001111
    123     1111011     111      1101111
    125     1111101      95      1011111
    127     1111111     127      1111111
    129    10000001     129     10000001
    131    10000011     193     11000001
    133    10000101     161     10100001
    135    10000111     225     11100001
    137    10001001     145     10010001
```

which clearly shows the bit-reversal at work for odd n. Even n are treated likewise, but as they end in one or more zeroes, these are rendered non-significant upon reversal, and do not show up in the output, for instance:

```
    n = 136 = 10001000 -> h(n) = 00010001 = 10001 = 17
```

Conclusion

Of course now that we fully know what h(n) does, we can then proceed to write a new, non-recursive, non-array implementation of h(n), like this:

```
    10 DEF FNF(N)=BVAL(REV$(BSTR$(N,2)),2)
```

That's all there's to it. This single-line (more like *half-line*) user-defined function will do the same as our previous recursive implementation (reversing the binary bits of its argument), only orders of magnitude faster (a mere hundredths of a second for any legal argument), and using orders of magnitude less memory (neither recursion nor arrays involved, though it does use BVAL and BSTR$, which are keywords from the Math ROM, and REV$ which is a very popular keyword featured in a number of LEX files).

Well, that's as far as it gets. I hope you've found all this interesting and enjoyable and perhaps even enlightening. Whatever the case, I'm sure you'll agree with me that having such a perfecty normal-looking recursive function definition actually perform a *bit-reversal* of its argument is quite *unexpected* and kind of surprising, to say the least !

As I see it, it is like this function, instead of changing the *arithmetic* value of its argument, it rather does alter its *shape* instead, its physical orientation in some sense, disregarding the actual value being involved.

Mathematics are so *beautiful*. And these wonderful HP machines can serve us well to sharpen our insights, our intuition, in this amazing journey.

As for the 'prize', *all of you* who have posted your ideas, programs and insights to this thread and wish to receive a copy of the 1280 x 1024 image in the mail, please send me an e-mail to my e-mail address listed here and I'll send it back to you at once.

Best regards from V.

Edited: 8 July 2004, 10:12 a.m.

I think, I missed the right way...

Posted by ***Tizedes Csaba [Hungary]*** *on* ***9 July 2004, 2:43 a.m.****, in response to S&SMC #7: My Solutions And Comments [LOOOOONG], posted by Valentin Albillo on 8 July 2004, 8:49 a.m.*

...but I try to find the series without recursion.

Cs.

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

Posted by ***Martin Cohen*** *on* ***9 July 2004, 2:50 p.m.****, in response to S&SMC #7: My Solutions And Comments [LOOOOONG], posted by Valentin Albillo on 8 July 2004, 8:49 a.m.*

This is a very cute recursion! How did you come up with it?

Also, you might submit it to the online encyclopedia of integer sequences (if it is not already there).

Martin Cohen

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Posted by **Valentin Albillo** on **12 July 2004, 5:34 a.m.**, in response to Re: S&SMC #7: My Solutions And Comments [LOOOOONG], posted by Martin Cohen on 9 July 2004, 2:50 p.m.*

Hi, Martin:

Martin posted:

"This is a very cute recursion! How did you come up with it?"

As they say, "magicians never tell"

"you might submit it to the online encyclopedia of integer sequences (if it is not already there)."

It is: Sequence A030101: Base 2 reversal of N (written in base 10)

```
ID Number: A030101
URL:       http://www.research.att.com/projects/OEIS?Anum=A030101
Sequence:  0,1,1,3,1,5,3,7,1,9,5,13,3,11,7,15,1,17,9,25,5,21,13,29,3,
           19,11,27,7,23,15,31,1,33,17,49,9,41,25,57,5,37,21,53,13,45,
           29,61,3,35,19,51,11,43,27,59,7,39,23,55,15,47,31,63,1,65,33,
           97,17,81,49,113,9,73,41,105,25,89,57
Name:      Base 2 reversal of n (written in base 10).
References Solutions to 17th USA Mat. Olympiad, Math. Mag., 62 (1989), 210-214
              (#3).
Links:     Michael Gilleland, Some Self-Similar Integer Sequences
Formula:   a(n) = 0, a(2n) = a(n), a(2n+1) = a(n) + 2^([log2(n)]+1). For n>0,
              a(n) = 2*A030109(n) - 1. - Ralf Stephan (ralf(AT)ark.in-berlin.de),
              Sep 15 2003
Program:   (PARI) a(n)=if(n<1,0,subst(Polrev(binary(n)),x,2))
See also:  Cf. A030109, A036044, A056539.
           Sequence in context: A000265 A040026 A093474 this_sequence A081432
              A060819 A089654
           Adjacent sequences: A030098 A030099 A030100 this_sequence A030102
              A030103 A030104
```

Thanks for your interest and best regards from V.

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

Posted by ***Carl Nelson*** *on* ***10 July 2004, 12:41 a.m.****, in response to S&SMC #7: My Solutions And Comments [LOOOOONG], posted by Valentin Albillo on 8 July 2004, 8:49 a.m.*

Here's a recursive solution for the HP-48G, probably also works on the derivative models as well. YMMV.

Note that the name of the object must match the references in the program. I had made a copy to make some modifications to, and missed one of the references, so I had a little co-recursion happening between two different objects. The amazing thing is that it worked. Well, until I deleted the older version. *Splat*

Also, I am curious here. Has anyone looked at how difficult this would be to implement as a recursive algorithm on a non-stack calculator? i.e. one of the TI calculators, not a higher-language beastie like the HP-71 and variants, that is.

In any case, here is my take on the function.

HP-48G recursive:

HOFN:

```
<< CASE DUP 1 == THEN END DUP 3 == THEN END DUP 4 MOD CASE DUP 1 == THEN
- 4 / DUP 2 * 1 + HOFN 2 * SWAP HOFN - END DUP 3 == THEN - 4 / DUP 2 * 1 + HOFN 3
* SWAP HOFN 2 * - END DROP 2 / HOFN END END >>
```

Re: S&SMC #7: My Solutions And Comments [LOOOOONG]

*Posted by **Carl Nelson** on **10 July 2004, 12:56 a.m.**, in response to Re: S&SMC #7: My Solutions And Comments [LOOOOONG], posted by Carl Nelson on 10 July 2004, 12:41 a.m.*

Let me try that again, with some spacing...

HP-48G recursive:

HOFN:

```
<< CASE
DUP 1 == THEN END
DUP 3 == THEN END
DUP 4 MOD
CASE
DUP 1 ==
THEN
- 4 / DUP
2 * 1 + HOFN 2 *
SWAP HOFN -
END
DUP 3 ==
THEN
- 4 / DUP
2 * 1 + HOFN 3 *
SWAP HOFN 2 * -
END
DROP 2 / HOFN
END
END
>>
```

Short & Sweet Math Challenges #8: Squares !

Posted by ***Valentin Albillo*** *on* ***20 Apr 2005, 10:09 a.m.***

Hi all, long time no see:

After many months have elapsed since I posted the last S&SMC, here's a new one which certainly lives up to its title. Should you feel like it, grab your favorite HP calculator(s) (any programmable one will do), then try and solve this

Challenge:

Find a **10-digit square** which consists of just **two different** numbers, from 1 to 9 (0's not allowed). For example, this would be a solution (featuring just the digits 1 and 3):

```
3113311313
```

if said number were a square which, alas, it isn't.

The solution is unique. You're expected to produce a **program** for your chosen calculator(s) that upon running will find a solution, also making sure there are no others. If you succeed, you may want to extend your program to also:

Find **all** 10-digit squares which consist of at most **three** different numbers, from 1 to 9 (0's not allowed). For example, this would be a solution, (featuring just the digits 3,5, and 7):

```
5733753373
```

if it were a square, which of course it isn't. There are 48 solutions in all.

As stated, you must produce a program for your HP calculator(s) that will compute and output the solution(s), the shorter and faster the better. Giving just the solutions or providing programs for machines other than HP calculators is very nice of you, thank you, but you simply forfeited the challenge, you loser ! ;-)

Next Friday I'll post the solutions and a 4-line program for a bare bones HP-71B which promptly finds out the solutions. Meanwhile be a sport and try your best. Remember, it's one thing to boast about your efficiency with RPN or RPL or whatever, and quite another to deliver the goods when challenged ... ;-)

Best regards from V.

Edited: 22 Apr 2005, 7:20 a.m. after one or more responses were posted

My brain hurts

*Posted by **Gene** on **20 Apr 2005, 10:39 a.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

:-)

C'mon, Gene ! It's quite simple :-)

*Posted by **Valentin Albillo** on **20 Apr 2005, 10:48 a.m.**, in response to My brain hurts, posted by Gene on 20 Apr 2005, 10:39 a.m.*

In your case I'll gladly make an exception: do post a program for a TI machine, should you feel more comfortable with it.

I know you're a keen defender of TI machines, just as I am of (some) SHARP ones, so this might be your opportunity to show that they're up to the task.

Best regards from V.

Re: C'mon, Gene ! It's quite simple :-)

*Posted by **Gene** on **20 Apr 2005, 10:51 a.m.**, in response to C'mon, Gene ! It's quite simple :-), posted by Valentin Albillo on 20 Apr 2005, 10:48 a.m.*

Insults? :-)

I like the old TI machines, but I am an RPN / RPL man!

I can't imagine trying to write a program to do this on a 100-step SR-56. :-)

Of course, that's primarily because I haven't thought through the problem enough yet, I suppose!

Gene

Re: C'mon, Gene ! It's quite simple :-)

Posted by ***Valentin Albillo*** *on* ***20 Apr 2005, 12:24 p.m.****, in response to Re: C'mon, Gene ! It's quite simple :-), posted by Gene on 20 Apr 2005, 10:51 a.m.*

Hi again, Gene:

Gene posted: *"Insults? :-)*

No, why on earth would you feel insulted ? Myself, I've stated a number of times that vintage SHARP machines are every bit as good if not better than contemporary HP ones and I wouldn't feel insulted in the least if someone were to offer me using a SHARP instead of an HP. But a TI ... that's a horse (donkey ?) of a different color ... come to think of it, you have a point, my apologies.

"I like the old TI machines, but I am an RPN / RPL man!"

I'm *not*.

"I can't imagine trying to write a program to do this on a 100-step SR-56. :-)"

I don't know about the SR-56, but I've got a quick'n'dirty 41CX version in some 60 steps ... Way to go.

Now stop ranting and do write a solution, man ! :-)

Best regards from V.

SR-56 memories...

Posted by ***Marcus von Cube*** *on* ***23 Apr 2005, 11:17 a.m.****, in response to Re: C'mon, Gene ! It's quite simple :-), posted by Gene on 20 Apr 2005, 10:51 a.m.*

> Quote:
>
> ---
>
> I can't imagine trying to write a program to do this on a 100-step SR-56.
>
> ---

My flag (re)setting algorithm will certainly not work! The most complicated problem I've ever tackled on the 56 was the solving of 3 linear equations with 3 unknowns. Because there are only 10 registers on the machine, I had to enter the constants on the right hand side of the equations 3 times (once for each unknown.)

And of course, I had to type in the whole program every time I wanted to use it...

... and a link to the HP-20S

*Posted by **Karl Schneider** on **23 Apr 2005, 1:37 p.m.**, in response to SR-56 memories..., posted by Marcus von Cube on 23 Apr 2005, 11:17 a.m.*

Marcus von Cube posted,

> Quote:
>
> ---
>
> The most complicated problem I've ever tackled on the [Texas Instruments TI-]56 was the solving of 3 linear equations with 3 unknowns. Because there are only 10 registers on the machine, I had to enter the constants on the right hand side of the equations 3 times (once for each unknown.)
>
> ---

The HP-20S (discontinued in 2002), also has only 10 storage registers. It includes a built-in loadable keystroke program for solving an exactly-determined 3x3 set of linear equations, using Cramer's Rule.

After the program "3by3" is loaded into program memory, the user enters the 9 matrix coefficients in registers 1-9, and the "upper" result constant into register 0. The remaining two constants are placed in the stack, separated by "INPUT". "XEQ A" calculates the solution.

The program can be used to solve an exactly-determined 2x2 system by entering the system as follows:

```
[a11 a12 0]  [b1]
[a21 a22 0]  [b2]
[ 0   0  1]  [ 0]
```

A matrix can be inverted by solving for $[1\ 0\ 0]^T$, $[0\ 1\ 0]^T$, and $[0\ 0\ 1]^T$ in succession.

Determinants can be calculated using "XEQ D" after loading the "3by3" program.

Kind of a pain in many respects, but it's better than nothing, I suppose...

-- KS

Re: Short & Sweet Math Challenges #7: Squares !

*Posted by **Larry Corrado, USA (WI)** on **20 Apr 2005, 9:03 p.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Valentin:

Rather than RPN or RPL, I used "whatever." Namely, Java on my Dell. Does that count? ;-)

I love my HP-25, which I used daily for 25 years. My HP-15C is beautiful, with its crisp, hi-contrast display. I love the glowing digits on my 34C... But when it comes to programming, I'll stick to Java or VB.NET.

I look forward to seeing your 4-line solution. Thanks for presenting the challange.

Ciao, Larry

Re: Short & Sweet Math Challenges #7: Squares !

*Posted by **Arnaud Amiel** on **21 Apr 2005, 4:40 a.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

This is short but dirty and slow (just over 1 hour) on the 49+. It will not work in RPN though...

```
<< RCLF {} 'RESULTS' STO 10. STWS BIN
1. 8. FOR I #1b I 1. + 9.
  FOR J 1. 1022.
    FOR K DUPDUP NOT ->STR TAIL OBJ-> DROP I * SWAP ->STR TAIL
          OBJ-> DROP J * + DUP
          IF FP THEN DROP ELSE 'RESULTS' STO+ END
    #1b +
    NEXT
  NEXT DROP
NEXT STOF
>>
```

And it tells me that 6661661161 is the only answer.

Now I have to find an other more efficient way to do it.

Arnaud

Erratum

*Posted by **Arnaud Amiel** on **21 Apr 2005, 5:53 a.m.**, in response to Re: Short & Sweet Math Challenges #7: Squares !, posted by Arnaud Amiel on 21 Apr 2005, 4:40 a.m.*

I forgot to type a SQRT (well a square root sign) just before the IF

Appart from that it works but so slowly... this is the kind of algorithm that would do well in assembly. Oh and I can't use this algorithm to go to step 2.

Arnaud

Edited: 21 Apr 2005, 6:21 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Jeff O.** on **21 Apr 2005, 6:34 p.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Without looking at Larry's or Arnaud's responses (I swear!) , in case they gave the answers, I'll go out on a limb and present the following:

> Quote:
>
> ---
>
> 1. Find a 10-digit square which consists of just two different numbers, from 1 to 9 (0's not allowed)....The solution is unique.
>
> ---

According to my hp-42S program, that number would be 6661661161, with a square root of 81619.

> Quote:
>
> ---
>
> 2. Find all 10-digit squares which consist of at most three different numbers, from 1 to 9 (0's not allowed).
>
> ---

According to my hp-42S program, the 10 digit squares that consist of **just** three different numbers are as follows:

```
COUNT   NUMBER  SQUARE
1       33334   1111155556
2       33335    1111222225
3       33338   1111422244
4       33359   1112822881
5       33989   1155252121
6       34641   1199998881
7       34827   1212919929
8       34965   1222551225
9       36361   1322122321
10      37962   1441113444
11      39388   1551414544
12      39581   1566655561
13      40204   1616361616
14      43923   1929229929
15      44623   1991212129
16      47162   2224254244
17      50235   2523555225
18      54123   2929299129
19      54335   2952292225
20      57665   3325252225
21      58167   3383399889
22      62156   3863368336
23      66515   4424245225
24      66665   4444222225
25      66667   4444488889
26      66668   4444622224
27      67485   4554225225
```

```
28      68187   4649466969
29      68962   4755757444
30      70107   4914991449
31      70173   4924249929
32      72285   5225121225
33      72335   5232352225
34      72475   5252625625
35      76478   5848884484
36      78196   6114614416
37      78541   6168688681
38      78881   6222212161
39      79162   6266622244
40      80408   6465446464
41      81346   6617171716
42      81404   6626611216
43      81813   6693366969
44      83666   6999999556
45      86478   7478444484
46      97773   9559559529
47      98336   9669968896
```

I was a bit distressed that I found only 47 of these, as apposed to the 48 that V. stated exist. However, upon reading his challenge, it does say *at most*. The group should therefore include those that have *just* 3 numbers listed above, plus the solution that has just 2 numbers, plus any that consist of just one number. A quick check of 111111111, 222222222, etc. with my calculator revealed that none of these are perfect squares. So the list of 47 above plus the unique solution to the first problem yields the 48 solutions to the second problem. (I hope.)
I'm quite sure that I used a very non-elegant, non-optimized, brute-force method. The program I wrote consisted of several basic parts.
1. create the 10 digit numbers by squaring all possible numbers that could create them. The maximum 10 digit number that meets the first criterion is 999999998, whose square root is 99999.99999. So no need to check anything above 99999. The minimum that satisfies the first criterion is 1111111112, whose square root is 33333.33333, so no need to check anything lower than 33334.
2. Break the 10 digit number up into 10 single digits stored in registers 1 through 10 (checking for numbers with zeroes and throwing them out along the way).
3. sort those digits in ascending order in registers 1 through 10.
4. run through registers 1 through 10 and count the number of times that two adjacent digits are not equal to each other.
5. See if the above is either 2, to solve the first problem or 3 to solve the second. If so, print out the number and the square.
6. Move on to the next number.

The program I wrote based on the above methodology was an HP-42S program. However, I developed and ran it on Free42. On my PC, my first version solved each of the problems in about 14 seconds. I then optimized the sorting and checking routines somewhat, and it solved each problem in about 9 seconds. I entered that version into a real 42S. After about 3 hours and 45 minutes I stopped it to make sure it was working correctly. It had found the first 8 solutions, and was about 2.4% of the way through the sequence of numbers. Based on that, it would have taken nearly 6 days to complete. Obviously, I chose a quite inefficient solution scheme. A different approach would be required to make it realistically soluable by a real 42S.

Edited: 25 Apr 2005, 7:16 a.m. after one or more responses were posted

Re: Short & Sweet Math Challenges #7: Squares !

Posted by ***Jonathan Puvis (New Zealand)*** *on* ***21 Apr 2005, 9:57 p.m.****, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Jeff O. on 21 Apr 2005, 6:34 p.m.*

I came up with pretty much the same algorithm (before you posted your message). I initially wrote it in Python (attached below), but my User-RPL version on the 48G ran for more than an hour after which i stopped it as the low battery warning had come one.

I could write the digit counting function in Saturn assembly to speed it up, but i'm sure there is a better algorithm for counting the digits, i just can't think of it.

```
#!/usr/bin/python
def count_digits(x):
  x = int(x)
  digits = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  while x:
    digits[x % 10] = 1
    x = x / 10
  if digits[0]:
    return 99
  return sum(digits)
for x in range(33333, 99999):
  sq = x * x
  if count_digits(sq) == 2:
    print '2:', x, sq
print
for x in range(33333, 99999):
  sq = x * x
  if count_digits(sq) <= 3:
    print '<=3', x, sq
```

HP-32S solution

*Posted by **Eamonn** on **21 Apr 2005, 11:32 p.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Valentin,

It's been a while since you posted a S&SMC! As ever, they are quite thought provoking.

I had to have a go at writing a program for the HP-32S to solve the first problem.

The basic idea was to generate all the possible ten-digit numbers that consist of two different numbers, take their square root and test to see if the fractional part is non-zero. There are C(9,8) * 2^10 = 36864 such numbers.

My first attempt would have taken about 10 hours to generate and test all such numbers (It generated and tested about 1 number per second). I won't bother posting it.

After thinking about it a bit, I took a different approach to generating the numbers that generates and tests all of them in about 27 minutes (almost 23 numbers per second).

This second approach uses almost all the 390 bytes of memory on the HP-32S. I'm guessing that it will also run on the HP-33S - not sure if it will be much faster or slower. Perhaps someone would be willing to try it out.

A different approach is needed for the second part of the problem - if my math is correct, there are almost 5 million possibilities for a 10-digit number made up of 3 unique digits.

Anyway, here is the program that I wrote. The program can be run by hitting XEQ V. It stops and displays each solution it finds. The user needs to hit the R/S key to continue after each solution. The number of solutions found is stored in the E register. It finds one solution for the first problem (6661661161 = 81619^2).

Eamonn

```
LBL T   ; Test the value that is in C
RCL C
SQRT
FP
X<>0?
RTN
1
STO+ E ; Count of how many solutions were found
RCL C
STO D  ; Store most recent solution
R/S    ; Stops to display latest solution - User needs to hit R/S to continue
RTN
[CHKSUM = A889]

LBL A
XEQ T
RCL J
STO+ C
XEQ T
RCL K
STO+ C
XEQ T
RCL J
STO+ C
```

```
XEQ T
RTN
[CHKSUM = 5BA1]

LBL B
XEQ A
RCL L
STO+ C
XEQ A
RCL M
STO+ C
XEQ A
RCL L
STO+ C
XEQ A
RTN
[CHKSUM = 3AA5]

LBL C
XEQ B
RCL N
STO+ C
XEQ B
RCL O
STO+ C
XEQ B
RCL N
STO+ C
XEQ B
RTN
[CHKSUM = D82D]

LBL D
XEQ C
RCL P
STO+ C
XEQ C
RCL Q
STO+ C
XEQ C
RCL P
STO+ C
XEQ C
RTN
[CHKSUM = 1A38]

LBL E
XEQ D
RCL R
STO+ C
XEQ D
RCL S
STO+ C
XEQ D
RCL R
STO+ C
XEQ D
RTN
[CHKSUM = EEA7]

LBL V ; <---- Program Entry Point
0
STO E ; Initialize number of answers
```

```
9
STO A ; First Digit
[CHKSUM = 8B6D]

LBL F
RCL A
1
-
x=0?  ; If B will be zero, then we're done
RTN   ; <---- Program Exit Point
STO B ; Second Digit
[CHKSUM = 60AB]

LBL G
11.019
STO i
1
STO J
[CHKSUM = 44BD]

LBL H ; Setting up constants used by the program
10
*
1
-
STO (i) ; K(i) = 10 * K(i-1) - 1
ISG i
GTO H
10.019
STO i
RCL A
RCL- B
[CHKSUM = AF37]

LBL I  ; Loop to multiply constants by (A-B)
STO* (i)
ISG i
GTO I
1111111111
RCL *B
STO C  ; C is the first value to test
XEQ E  ; Call recursive generation of test values
DSE B  ; Loop on second digit
GTO G
DSE A  ; Loop on first digit
GTO F
[CHKSUM = 62F8]
```

Re: HP-32S solution

*Posted by **Arnaud Amiel** on **24 Apr 2005, 3:14 a.m.**, in response to HP-32S solution, posted by Eamonn on 21 Apr 2005, 11:32 p.m.*

Quote:

> This second approach uses almost all the 390 bytes of memory on the HP-32S. I'm guessing that it will also run on the HP-33S - not sure if it will be much faster or slower. Perhaps someone would be willing to try it out.

It takes about 19min on the 33s that is about 33% faster.

Arnaud

Re: Short & Sweet Math Challenges #7: Squares !

*Posted by **Bram** on **22 Apr 2005, 7:03 a.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Hi V,

1. Thanks for the challenge

2. Did you loose track? Think there already has been a #7
http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv014.cgi?read=59918

3. My program is for the 32SII. After a short analysis it appeared that "only" the numbers between 31622 and 99999 are eligible for an answer.I check them all for the criteria and I break out the moment a criterion fails. That's the only speed up, so finding the answer 81619 takes several hours.
The program is for the 3 dupl. problem; if you remove the marked instructions, you get the program for the single solution problem.

```
LBL A   ; CK=1BB7 012.5
31622
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C
RCL N
99999
x>=y?
GTO B
STOP

LBL C   ; CK=BA5C 013.5
FS? 0
PSE
CF 1
CF 2
CF 3    ; <=
STO R
10      ; squared number consists of 10 digits
STO I
LBL J   ; CK=C11A 021.0
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed, next number
FS? 1   ; digit already encountered?
GTO K   ; yes
STO X   ; no, store it for comparison with more digits
SF 1    ; digit now has encountered
GTO M
LBL K   ; CK=CDA4 015.0
```

```
RCL X   ; get first unique digit
x=y?    ; equal to the one in question?
GTO M
x<>y
FS? 2   ; like before, but now for the second digit
GTO L
STO Y
SF 2
GTO M
LBL L   ; CK=B38A 015.0
RCL Y
x=y?
GTO M
x<>y    ; <=
FS? 3   ; <=    like before, but now for the third digit
GTO N   ; <=
STO Z   ; <=
SF 3    ; <=
GTO M   ; <=
LBL N   ; CK=2764 007.5
RCL Z   ; <=
x=y?    ; <=
GTO M   ; <=
RTN
LBL M   ; CK=102A 010.5
DSE I
GTO J   ; go to the next digit to examine
RCL N   ; all digits examined, so now we have an answer
x²
STOP
RTN


total length:     119.5

;        storage registers
;        N      number to examine
;        R      number² being split into digits
;        X Y (Z) the digit to occur
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        JKL(N) place holders
;        M      increment to the next digit to examine
;
;        flags
;        0      set => show progression
;        12(3)  keep track of new occurrance of digit
```

Thanks a lot, Bram !

*Posted by **Valentin Albillo** on **22 Apr 2005, 7:25 a.m.**, in response to Re: Short & Sweet Math Challenges #7: Squares !, posted by Bram on 22 Apr 2005, 7:03 a.m.*

Hi, Bram:

Bram posted:

"2. Did you loose track? Think there already has been a #7 http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv014.cgi?read=59918 "

Yes, I certainly did. Though I searched my files to see what the current number should be, it seems I didn't keep a record of that particular S&SMC (#7), so I thought that #6 was the very last to date.

I've duly corrected the numbering, so that it won't be kept in the MoHP Archives section with the wrong, duplicated number.

Thanks a lot, see my post with the solutions and comments within 12 hours.

Best regards from V.

Re: Thanks a lot, Bram !

*Posted by **Bram** on **22 Apr 2005, 10:28 a.m.**, in response to Thanks a lot, Bram !, posted by Valentin Albillo on 22 Apr 2005, 7:25 a.m.*

Anytime Valentin, and in the meantime I rewrote my program to a more versatile one based on indexing. The marked line defines the criterion for the number of digits. Change the 3 into 2 for the first problem. Changing the comparison in the next line is for adoption to exactly x different digits or a least y different digits a.s.o.

regards,
Bram

```
LBL A   ; CK=1BB7 012.5
31622
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C
RCL N
99999
x>=y?
GTO B
STOP

LBL C   ; CK=78F9 009.0
FS? 0
PSE
STO R
9
STO i
LBL I   ; CK=DD00 010.5
0
STO (i)
DSE i
GTO I
10      ; squared number consists of 10 digits
STO J
LBL K   ; CK=0B4E 027.0
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed, next number
10
*
STO i
1
STO+ (i)
DSE J
GTO K
10
STO i
LBL J   ; CK=E621 022.5
```

```
RCL (i)
x#0?
1
STO+ J
DSE i
GTO J
3       ; <= max. number of different digits
RCL J
x>y?
RTN
RCL N
x²
STOP
RTN


total length:     106.0

;        storage registers
;        N      number to examine
;        R      number² being split into digits
;        A-I&J  for counting occurrences
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        IJK    place holders
;
;        flags
```

Re: Third version

*Posted by **Bram** on **26 Apr 2005, 4:44 p.m.**, in response to Re: Thanks a lot, Bram !, posted by Bram on 22 Apr 2005, 10:28 a.m.*

Hi Valentin, once more....

My second program was more versatile, but much slower than my first one. I tried to speed up my original program by squeezing the inner part to a minimum number of instructions. In only 15 lines (starting at ENTER) I test all criteria and you may be interested in how I use "conditional comparison" to get this done.
I hope things are clear enough by my comments in the program.

```
LBL A   ; CK=1BB7 012.5
31622   ; smallest root of ten digits number minus 1
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C   ; examine all ten digits squares
RCL N
99999
x>=y?
GTO B
STOP

LBL C   ; CK=AB00 015.0
FS? 0   ; wish to see progression?
PSE     ; yes, show square that's going to be examined
STO R
0
STO X   ; clear x Y Z on behalf of the three different digits
STO Y
STO Z   ; <=
10      ; squared number consists of ten digits
STO I
LBL J   ; CK=9B2F 037.5
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed => next number
ENTER   ; put the digit to examine into Y for comparisons
x<>X    ; save it in X and get former value from X
x#0?    ; if former value was 0, we have a new digit; goto M for next digit
x=y?    ; if equal, we have a duplicate digit; goto M for next digit
GTO M
x<>X    ; digit is a new, second one; restore X with original contents
        ; and get back the digit to examine into X
x<>Y    ; same as before; check against 2nd distinct digit that has occurred
x#0?
x=y?
GTO M
x<>Y    ; <= C
x<>Z    ; <=  L
x#0?    ; <=   E
```

```
x=y?    ; <=    A
GTO M   ; <=     R these lines when checking for just two different digits
RTN     ; we get to this point when a fourth digit is reached => next number
LBL M   ; CK=102A 010.5
DSE I
GTO J   ; go to the next digit to examine
RCL N   ; all digits examined, so now we have an answer
x²      ; proudly show the result
STOP    ; and allow the user to see it
RTN     ; continue with next number


total length:     100.0

;        storage registers
;        I      loop counter for ten digits
;        N      number to examine
;        R      number² being split into digits
;        X Y (Z) the digit to occur
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        J      place holder
;        M      increment to the next digit to examine
;
;        flags
;        0      set => show progression
```

regards,
   Bram

Re: Short & Sweet Math Challenges #8: Squares !

Posted by ***Olivier De Smet*** *on* ***22 Apr 2005, 9:51 a.m.****, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Here is a quick hack for an HP86B (with advance prog module)

```
10 l=0 @ o=3 @ t=TIME  @ FOR n=33333 TO 99999 @ IF o=10 THEN o=0 @ GOTO 60
20 k=1 @ SFLAG "" @ m=n*n @ SFLAG m MOD 10 @ m=m\10 @ FOR j=2 TO 10 @ b=m MOD
10 @ IF b=0 THEN 60
30 IF FLAG (b) THEN 50
40 IF k=3 THEN 60 ELSE SFLAG b @ k=k+1
50 m=m\10 @ NEXT j @ l=l+1 @ DISP l,n,n*n
60 o=o+1 @ NEXT n @ DISP "Fini:";TIME -t @ END

run
 1                     33334                 1111155556
 2                     33335                 1111222225
 3                     33338                 1111422244
 4                     33359                 1112822881
 5                     33989                 1155252121
 6                     34641                 1199998881
 7                     34827                 1212919929
 8                     34965                 1222551225
 9                     36361                 1322122321
 10                    37962                 1441113444
 11                    39388                 1551414544
 12                    39581                 1566655561
 13                    40204                 1616361616
 14                    43923                 1929229929
 15                    44623                 1991212129
 16                    47162                 2224254244
 17                    50235                 2523555225
 18                    54123                 2929299129
 19                    54335                 2952292225
 20                    57665                 3325252225
 21                    58167                 3383399889
 22                    62156                 3863368336
 23                    66515                 4424245225
 24                    66665                 4444222225
 25                    66667                 4444488889
 26                    66668                 4444622224
 27                    67485                 4554225225
 28                    68187                 4649466969
 29                    68962                 4755757444
 30                    70107                 4914991449
 31                    70173                 4924249929
 32                    72285                 5225121225
 33                    72335                 5232352225
 34                    72475                 5252625625
 35                    76478                 5848884484
 36                    78196                 6114614416
 37                    78541                 6168688681
 38                    78881                 6222212161
 39                    79162                 6266622244
 40                    80408                 6465446464
 41                    81346                 6617171716
 42                    81404                 6626611216
 43                    81619                 6661661161
 44                    81813                 6693366969
 45                    83666                 6999999556
 46                    86478                 7478444484
 47                     97773                   9559559529
 48                     98336                   9669968896
Fini: 108.019
•
```

In fact as 0 isn't allowed only integer from 33333 to 99999 are tested and not ending with 0.

The given program is for the second problem (which include the first). Need 1h50 on real hardware to find all solutions. Change line 40 with '40 IF k=2 ...' for the first problem.

Thanks again for the challenge.

Olivier

Edited: 22 Apr 2005, 9:57 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

Posted by ***Olivier De Smet*** *on* ***22 Apr 2005, 11:16 a.m.****, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Olivier De Smet on 22 Apr 2005, 9:51 a.m.*

A better solution for the first problem:

```
list
10 t=TIME
20 FOR i=0 TO 511 @ a=1000000000+VAL (DTB$ (i)) @ b=VAL (DTB$ (511-i))
30 FOR j=1 TO 9 @ FOR k=1 TO 9 @ m=j*a+k*b @ IF FP (SQR (m))=0 THEN DISP SQR
(m),m
40 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 517068
•
run
 81619                6661661161
Fini: 19.118
```

Just 20 mins for finding the value, but need the I/O module.

Olivier

Re: Short & Sweet Math Challenges #8: Squares !

Posted by ***J-F Garnier*** *on* ***22 Apr 2005, 11:26 a.m.****, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Olivier De Smet on 22 Apr 2005, 11:16 a.m.*

Hi Olivier,

I was just about to post a similar solution for the 71B. You can improve this solution by noticing that only numbers ending by 1, 4, 5, 6 or 9 can be a square number. But I didn't find how to solve the 3-digit problem with this approach.

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Olivier De Smet** on **22 Apr 2005, 11:35 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by J-F Garnier on 22 Apr 2005, 11:26 a.m.*

Yes I know but searching to eliminate such non solution is too costly in time :)

A better one :

```
list
10 t=TIME
20 FOR i=0 TO 511 @ a=1000000000+VAL (DTB$ (i)) @ b=VAL (DTB$ (511-i)) @ FOR
j=1 TO 9 @ m=j*a @ FOR k=1 TO 9 @ m=m+b @ IF FP (SQR (m))=0 THEN DISP SQR
(m),m
30 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 517065
run
 81619                6661661161
Fini: 15.199
•
```

Only 16 mins :)

Olivier

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **J-F Garnier** on **24 Apr 2005, 11:00 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Olivier De Smet on 22 Apr 2005, 11:35 a.m.*

A slighly improved version of Olivier's solution, for the HP-71B and using the BSTR$ function from the Math module:

```
10 T=TIME
15 OPTION BASE 1 @ DIM F(5)
16 DATA 1,4,5,6,9
17 READ F()
20 FOR I=0 TO 511 @ A=VAL(BSTR$(I,2))*10+1 @ B=VAL(BSTR$(511-I,2))*10
30 FOR J=1 TO 5 @ M=F(J)*A @ FOR K=1 TO 9 @ M=M+K*B @ IF FP(SQRT(M))=0 THEN
DISP SQRT(M),M
40 NEXT K @ NEXT J @ NEXT I @ DISP "Fini:";TIME-T @ END
 81619                6661661161
Fini: 45.77
```

Running on Emu71 using a 1.7GHz processor.

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Olivier De Smet** on **25 Apr 2005, 3:02 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by J-F Garnier on 24 Apr 2005, 11:00 a.m.*

Hi,

As I told you I don't need to optimize, my programm need only 15.2 sec running on an HP86B emulator on a 1.8Ghz pentium M :)

But you're right it's faster like this :

```
list
10 t=TIME  @ OPTION BASE 0@ DIM f(4)
20 f(0)=1 @ f(1)=4 @ f(2)=5 @ f(3)=6 @ f(4)=9
30 FOR i=0 TO 511 @ a=VAL (DTB$ (i))*10+1 @ b=VAL (DTB$ (511-i))*10 @ FOR j=0
TO 4 @ m=a*f(j) @ FOR k=1 TO 9 @ m=m+b @ IF FP (SQR (m))=0 THEN DISP SQR
(m),m
40 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 516894
run
 81619                 6661661161
Fini: 8.486
•
```

Only 8.846 sec :)

Olivier

Edited: 25 Apr 2005, 6:02 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Olivier De Smet** on **22 Apr 2005, 11:39 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by J-F Garnier on 22 Apr 2005, 11:26 a.m.*

For 3 digit as someone said it there is too much candidates, so the reverse approach should be better I think because you only have to test (99999-33333) candidates.

For 2 digits it's the cost of the test which is important as you have either 66666 or 512*9*9 candidates.

Olivier

Hi, Jean-François !

*Posted by **Valentin Albillo** on **22 Apr 2005, 11:46 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by J-F Garnier on 22 Apr 2005, 11:26 a.m.*

Hi, J-F:

Your wonderful Emu71 HP-71B's emulator runs my solution program for the HP-71B in under 40 seconds !

By the way, did you get to see my latest HP-71B articles published in Datafile ? All of the featured programs were created with the help of your Emu71, and it runs them like a charm. I do include footnotes recommending both your emulator and Hrastprogrammer's one for the 48/49, so that readers can know that they can try the programs and fully enjoy them themselves.

The ones featured in the latest Datafile issue are an extremely good test of your emulator's performance, as they make use not only of the emulated basic 71B itself, but also of the emulated Math and HP-IL ROMs as well, and it does certainly pass the taxing ordeal with flying colors, to say the least !

Best regards from V.

Re: Hi, Valentin

*Posted by **J-F Garnier** on **24 Apr 2005, 9:52 a.m.**, in response to Hi, Jean-François !, posted by Valentin Albillo on 22 Apr 2005, 11:46 a.m.*

>> By the way, did you get to see my latest HP-71B articles published in Datafile ?

Unfortunatly no, I (still) didn't subscribe to Datafile, shame on me ...

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Marcus von Cube** on **22 Apr 2005, 1:36 p.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

This is a solution for the HP 42S and HP 41C. [Edited]

Usage:

Enter the number of allowed different digits (2 or 3) in X and XEQ "SQ10". After the 33333 is displayed you have to press R/S (You can correct the starting number, if you like.)

If you enter a number >9, XEQ "SQ10" will just run the comparison engine (LBL 00) and show "OK" if the criterion is met. You need to store the number of digits manually in R02 before you can use the program this way.

```
LBL "SQ10"   Main entry point
0
STO 09
Rv
9
X<>Y
X<=Y?
GTO 10
XEQ 00
RCL 00
FC? 00
RTN
"OK"
AVIEW
RTN
```

Now comes the checking engine. It works on flags 0-9 which are all set in the beginning and cleared indirectly for each corresponding digit. The reset operations are counted in R00. If the count reaches the limit in R02, the check has failed and flag 0 will be cleared. (This is done at no cost, if a "0" digit is found.) A cleared flag 0 stops the loop.

R09 counts the total number of calls for statistical purposes.

```
LBL 00
VIEW ST X
SF 00
SF 01
SF 02
SF 03
SF 04
SF 05
SF 06
SF 07
SF 08
SF 09
0
STO 00
1
STO+ 09
Rv                RDN for HP 41
Rv

LBL 01            digit loop
STO 01
10
```

```
MOD
FC?C IND ST X     this does the flag testing and clearing!
GTO 02            this digit is already counted
RCL 00            get count
RCL 02            max # of different digits allowed
X<=Y?
CF 00
FC? 00
RTN               too many different digits
1
STO+ 00           count
LBL 02            get next digit
RCL 01
10
/
IP                INT for HP 41
X!=0?             any digits left?
GTO 01
RTN

LBL 05            find smallest digit from flags
                  works only after a passed check!
RCL 00
RCL 02
-
X=0?              maximum # of digits reached?
GTO 06            look for smallest
Rv
1                 return 1, because less than the
RTN               allowed number of digits were present

LBL 06
1
+
FS? IND ST X
GTO 06
RTN
```

This is the controlling loop. It starts at 33333+1 but it does not try each and every number up to 99999. When the left most 4 digits do not pass the check, there is no need to try any number that would result in the same 4 digits. I simply construct a number with the next higher value and get the SQRT from it:

If guess^2 == p,qrs,tuv,wxy. then new guess = SQRT((p,qrs.+X)*10e6 + Y11,111.)

p,qrs+X is determined by adding 1 in a loop and checking the resulting 4 digit number until the check passes. Y is the smallest digit found in p,qrs+X by the check routine.

A similar trick is used to skip guesses when the first 3 digits fail the check.

(The following is awful spaghetti code and I'm willing to streamline it when I have time.)

```
LBL 10
STO 02            store # of allowed digits
0
STO 05
STO 06            scratch registers for first 3 to 4 digits
33333             the loop starts with this initial guess
STOP              correct the initial guess at will
                  (The loop starts with this number + 1!)

LBL 11
STO 03            store first guess - 1
```

```
LBL 12
1
STO+ 03             increment guess
999999              final value
RCL 03
X>Y?
GTO 20              no more numbers to try
X^2
STO 04              store result to check (for later)
1E6
/
IP                  first 4 digits (INT for HP 41)
x<>Y
RCL 05              first 4 digits from last loop
X!=Y?               changed?
GTO 13              check and find best next guess
RCL 04              squared guess
XEQ 00              check it
FC? 00
GTO 12              failed, try next
CLD                 OK, show result
RCL 03
RCL 04
BEEP
STOP
GTO 12              back to main loop

LBL 13              find the next guess quicker by checking
1                   the first 3 or 4 digits only
-
STO 05

LBL 14
1
STO+ 05             new value for first 4 digits
RCL 06              first 3 digits (saved)
RCL 05              first 4
10
/
IP
X=Y?
GTO 19              the first 3 are the same as before

LBL 15
STO 06              new value for first 3
XEQ 00              check
FS? 00
GTO 16              first 3 digits are OK, must try first 4
RCL 06
1
+
GTO 15

LBL 16              new first 4 digits from first 3 digits
RCL 06
10
*
XEQ 05              find smallest digit from flags
+
STO 05

LBL 19              check the first 4 digits
```

```
RCL 05
XEQ 00            check
FC? 00
GTO 14            check failed, try next
RCL 05            compute new guess
10
x
XEQ 05            find smallest digit
+
1E5
x
11110
+
SQRT
IP
GTO 11            restart with new guess

LBL 20            finis
CLD
CLX
END
```

See next post for some statistics.

Edited: 23 Apr 2005, 2:33 p.m. after one or more responses were posted

Some Statistics for the above

*Posted by **Marcus von Cube** on **23 Apr 2005, 11:09 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Marcus von Cube on 22 Apr 2005, 1:36 p.m.*

Performance:

The 42S took about 90 minutes to find the 2 digit solution. My 41CX arrived after 4 hours and 10 minutes.

The number of checks performed was 4487 until the first solution (81619) and 6625 total. (These numbers were computed before I've made my last changes, but the results should be similar.)

Anybody for a faster check routine? It looks like the loop over the digits takes most of the computing time.

Re: Some Statistics for the above

*Posted by **Marcus von Cube** on **24 Apr 2005, 8:28 a.m.**, in response to Some Statistics for the above, posted by Marcus von Cube on 23 Apr 2005, 11:09 a.m.*

With the current version I could reduce the time and number of calls further:

3771 calls to the check routine and 75 minutes to solve the 2 digit problem on a real unmodified 42S. The 41C took 3:45 minutes. (It's funny to watch the flags toggle in the display :-))

The total number of calls to the checking routine was 5140. This will be significantly higher for the three digit problem, because less numbers can be ruled out beforehand based on the first 4 digits.

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Arnaud Amiel** on **23 Apr 2005, 6:47 p.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

As stated above, I believed that assembly was more suited to this problem so I gave it a try. The 49G and 49g+ are the only calcs to have assembly available out of the box so I tried on the 49g+. The program below (non optimised) solves question 2 in 43s. A 49g would take 1min 43s. It is easy to port it on the 48 where it should run as fast as on a 49 (or half speed on the SX).

Here is the program

```
SAVE
SETDEC
LC 33332
R1=C.A

*NextNumber
C=R1.A C+1.A SKIPNC { P=0 SETHEX LOADRPL }
R1=C.A
C=0.W C=R1.A
CSL.W CSL.W CSL.W
A=C.W
GOSBVL =SPLITA
C=B.W D=C.W C=A.W
GOSBVL =MULTF
GOSBVL =PACK
C=0.W C=A.M
CSR.W CSR.W CSR.W CSR.W CSR.W

P=10 A=0.S B=0.S D=0.S
*TestNumber
P-1 GOC GoodNumber
CSRC
?C=0.S GOYES NextNumber
?A#0.S { A=C.S GOTO TestNumber }
?A=C.S GOYES TestNumber
?B#0.S { B=C.S GOTO TestNumber }
?B=C.S GOYES TestNumber
?D#0.S { D=C.S GOTO TestNumber }  %remove for question 1
?D=C.S GOYES TestNumber           %remove for question 1
GOTO NextNumber

*GoodNumber
A=0.W A=R1.A ASRC ASRC ASRC ASRC ASRC ASRC  A+4.A
GOSBVL =PUSH%
SAVE
SETDEC
GOTO NextNumber
```

As usual archive before playing with assembly

Arnaud

PS: If you want I can send you the compiled program for 48 or 49 by email

[LONG] S&SMC #8: My original solutions, comments and curiosities !

*Posted by **Valentin Albillo** on **25 Apr 2005, 5:13 a.m.**, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

Hi all,

Thanks to all of you interested in my S&SMC #8 and most specially to those who provided such ingenuous and elegant solutions for such a variety of HP calcs and computers, we really saw a lot of interesting techniques being used here for the one and only purpose: find the solutions and find them fast. Congratulations to all of you, I *really* feel most satisfied with your contributions !

As promised, this is my original, 4-line solution for the HP-71B:

```
10 C=0 @ FOR N=33334 TO 99999 @ M$=STR$(N*N) @ IF POS(M$,"0") THEN 40 ELSE
T$=M$[1,1]
20 FOR I=2 TO 10 @ IF POS(T$,M$[I,I]) THEN 30 ELSE IF LEN(T$)=3 THEN 40 ELSE
T$=T$&M$[I,I]
30 NEXT I @ C=C+1 @ DISP N;M$,T$
40 NEXT N @ DISP "OK:";C;"found"
```

Calling it produces the 48 solutions, displaying the number, its square, and the distinct digits it consists of:

```
>CALL

 33334 1111155556    156
 33335 1111222225    125
 33338 1111422244    142
 33359 1112822881    128
 33989 1155252121    152
 34641 1199998881    198
          ...
 81404 6626611216    621
 81619 6661661161    61    <- only *two* different digits, the solution to
the first part
 81813 6693366969    693
 83666 6999999556    695
 86478 7478444484    748
 97773 9559559529    952
 98336 9669968896    968

OK:  48  found
```

It takes 3 min 28 sec when run under Emu71 in a 1.4 Ghz PC. The program searches all numbers which generate 10-digit squares fulfilling the conditions. As no 0's are allowed, the smaller possible square would be 1111111111, so we begin the search at SQR(1111111111) rounded to the nearest integer, which is 33334, and go on searching till SQR(9999999999), which comes out to 99999. We square each number in turn and convert the result to a string. The first POS then immediately discards numbers having a "0" anywhere. Then we keep on extracting individual digits which are added to an auxiliary string, using POS to detect repetitions (in which case we don't add the digit again) and LEN to see if there are more then 3 distinct digits already collected, in which case we forfeit further processing of this number and go instead to check the next candidate.

However, this is not the fastest way to proceed because the HP-71B is optimized for numerical computations and string processing is relatively slow. So I originally came out instead with this other version, which uses flags to register which digits have been used:

```
10 C=0 @ FOR N=33334 TO 99999 @ M=N*N @ CFLAG ALL @ P=0
20 FOR I=1 TO 10 @ D=MOD(M,10) @ IF NOT D THEN 50 ELSE M=M DIV 10
30 IF FLAG(D) THEN 40 ELSE IF P=3 THEN 50 ELSE SFLAG D @ P=P+1
40 NEXT I @ C=C+1 @ DISP N;N*N
50 NEXT N @ DISP "OK:";C;"found"
```

It works the same, only using flags instead of strings. Each candidate number is decomposed into its constituent digits. If the digit is a 0, we skip to the next candidate, else a flag is set corresponding to the digit. If more than 3 flags have alrady been set, no further digits are isolated but we go instead to test the next candidate. This version finds the exact same solutions but it takes just 72 seconds (under Emu71), i.e., it's 3 times faster than the string version. A real HP-71B takes much longer of course, but it's a real beauty to see the display's flag indicators making a frantic dance while the program runs !

In both versions, changing the "3" to any other single digit N (1-9) would find squares made up of N distinct digits. For instance, with N=2, it produces 6661661161, the only 10-digit square consisting of just two different digits, 1 and 6 in this case. Also, you can search for 12-digit squares instead, say, by simply changing the search limits to 333334 and 999999 respectively, and by changing the limit of the I loop to 12. This can be automated by simply creating a new, generalized version that asks the user for the number of digits and maximum different values, like this:

```
 1 INPUT "# Digits (1-12) = ";U @ INPUT "# Diff. val. (1-9) = ";V
10 C=0 @ FOR N=INT(SQR((10^U-1)/9)) TO INT(SQR(10^U-1)) @ M=N*N @ CFLAG ALL
20 P=0 @ FOR I=1 TO U @ D=MOD(M,10) @ IF NOT D THEN 50 ELSE M=M DIV 10
30 IF FLAG(D) THEN 40 ELSE IF P=V THEN 50 ELSE SFLAG D @ P=P+1
40 NEXT I @ C=C+1 @ DISP N;N*N
50 NEXT N @ DISP "OK:";C;"found"
```

Let's try it ! To find the five 5-digit squares consisting of at most 2 different values:

```
>RUN

# Digits (1-12)    = 5  [ENTER]
# Diff. val. (1-9) = 2  [ENTER]

 109  11881
 173  29929
 212  44944
 235  55225
 264  69696

OK: 5 found
```

While to find the thirty 12-digit squares consisting of at most 3 different values:

```
>RUN

# Digits (1-12)    = 12 [ENTER]
# Diff. val. (1-9) = 3  [ENTER]

 333334  111111555556
 333335  111112222225
 333338  111114222244
 333359  111128222881
 333858  111461164164
 363639  132233322321
 387288  149991994944
 461761  213223221121
 492162  242223434244
 505525  255555525625
 515415  265652622225
 586893  344443393449
 649788  422224444944
 658357  433433939449
 664388  441411414544
 665908  443433464464
 666515  444242245225
 666665  444442222225
 666667  444444888889
 666668  444446222224
 682908  466363336464
 782104  611686666816
 805262  648446888644
 818333  669668898889
 828667  686688996889
 834386  696199996996
 869924  756767765776
 939938  883483443844
 974417  949488489889
 994836  989698666896

OK: 30 found
```

The 'flags' version also has the added advantage that it translates very easily to RPN machines with little or no alpha capabilities, such as the HP-41C or HP-15C. For instance, here's the translated version for an HP-41CX, a meager 46 steps:

```
01 *LBL "SQ3"  16  STO 04      31  GTO 04
02  CLX        17   10         32 *LBL 05
03  X<>F       18  STO 05      33   1
04  CF 08      19 *LBL 01      34  ST+ 02
05  CF 09      20  RCL 03      35   1E5
06  RCLFLAG    21  INT         36  RCL 02
07  STO 01     22   10         37  X#Y?
08  33334      23  ST/ 03      38  GTO 00
09  STO 02     24  MOD          39  STOP
10 *LBL 00     25  X=0?        40 *LBL 04
11  X^2        26  GTO 05      41  DSE 05
12  STO 03     27  FS? IND X   42  GTO 01
13  RCL 01     28  GTO 04      43  RCL 02
14  STOFLAG    29  SF IND X    44  X^2
15  .003       30  ISG 04      45  VIEW X
                                46  GTO 05
```

It runs under the V41 emulator in 72 min (a real HP-41CX would take a lot longer) to produce all 48 solutions, and again, it's a joy to see all those flag indicators frantically turning on and off:

```
[R/S]

 1111155556
 1111222225
    ...
 9559559529
 9669968896
```

For practical reasons and in order to to make it accessible to most HP calculators, this challenge specifically addressed 10-digit squares, but if you feel like it there are marvels out there waiting to be discovered, here are some examples:

```
    10099510939154979751^2 = 102000121210111101102120011101220022001 (39
digits, all 0,1,2)

   557963558954625926861^2 = 311323333121312322332133323111223321313321 (42
digits, all 1,2,3)

6757548110569887742949784^2 = 456644564666666555445565455644644555565545646656
(48 digits, all 4,5,6)
```

Actually, even the 48-item list of solutions for the original challenge holds a number of very interesting results upon close inspection, such as these beautifully-patterned squares:

```
  1111155556  ( 11111-5555-6    )
  1111222225  ( 1111-22222-5    )
  1616361616  ( 16-16-36-16-16  )
  2929299129  ( 29-29-29-91-29  )
  4444222225  ( 4444-22222-5    )
  4444488889  ( 44444-8888-9    )
  5252625625  ( 52-52-625-625   )
  6617171716  ( 66-17-17-17-16  )
```

Thanks for your interest in this humble S&SMC #8, hope you enjoyed it, and

Best regards from V.

Posted by ***Tizedes Csaba [Hungary]*** *on* ***27 Apr 2005, 7:33 a.m.****, in response to Short & Sweet Math Challenges #8: Squares !, posted by Valentin Albillo on 20 Apr 2005, 10:09 a.m.*

:) Err, I missed it...! Thanks for the challenge! Csaba

Posted by ***Valentin Albillo*** *on* ***27 Apr 2005, 9:30 a.m.****, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Tizedes Csaba [Hungary] on 27 Apr 2005, 7:33 a.m.*

Best regards from V.

Re: Math Challenge #8:- HP-16C Version

Posted by ***Bill (Smithville, NJ)*** *on* ***29 Apr 2005, 1:14 p.m.***

Hi Valentin,

I have always enjoyed your Math Challenges but until now have never actually tried programming one. When I first read this one, I starting thinking that some sort of bit shifting could be used to solve it. So I got my HP-16C out, loaded it with new batteries and sit down last night with the manual to refresh my memory of how it operated and could it be used to solve this. I came up with the following 84 step program:

```
001     LBL A
002     DEC     set to decimal
003     34
005     WSIZE   set wordsize to 34 bits
006     33333   starting number
011     STO 0
012     LBL B   main loop from 33333 to 999999
013     RCL 0
014     1
015     +       add one and save number
016     STO 0
017     99999
022     X <> Y
023     X > Y?
024     GTO E   all done
025     0
026     STO 4   Initilize R4 to zero bits
027     RCL 0
028     ENTER
029     *
030     STO 1   store square of number
031     10
033     STO I   store index
034     LBL C   loop over 10 digits
035     RCL 1
036     RCL 1
037     10
039     /
040     10
042     *
043     -       right most digit
044     X = 0?
045     GTO B   if zero, then exit this number
046     STO 2   save digit
047     10
049     STO 3
050     LBL D   loop 9 to 1 to see which digit it is
051     RCL 3
052     1
053     -
054     STO 3
055     RCL 2
056     XOR
057     X NE 0?
058     GTO D   not this digit, go try next
059     1
060     RCL 3
061     RL N
062     RCL 4
```

```
063     OR      set bit in R4 to match digit found
064     STO 4
065     RCL 1
066     RCL 2
067     -
068     10
070     /       strip off right most digit
071     STO 1
072     DSZ
073     GTO C   go try next digit
074     RCL 4   all digits scanned, recall bit register
075     #B      count number found
076     3
077     X NE Y?
078     GTO B   if not exactly 3, then go try next number in sequence
079     RCL 0
080     R/S     answer found - stop and display
081     GTO B
082     LBL E   all done
083     0
084     R/S     stop and display zero
```

I've run it this morning and it found the first 4 solutions in a little over one hour. It's still working on finding the next solution.

I'm sure the program could be optimized, since it's been 20 years since I have actually programmed with it.

Thanks to you Valentin, I have had a lot of fun getting re-aquainted with an old friend, the HP-16C.

Bill

Re: Math Challenge #8:- HP-16C Version

*Posted by **Don Shepherd** on **30 Apr 2005, 10:58 a.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Bill (Smithville, NJ) on 29 Apr 2005, 1:14 p.m.*

Hey Bill, that's great work on the venerable 16c. You said you found 4 solutions, I thought that Valentin said that there was only one solution. I'm going to look at your code to get some insights myself. I think I'll try to write a program for the 12cp to solve it (since it's faster than the older voyagers).

Re: Math Challenge #8:- HP-16C Version

*Posted by **Bill (Smithville)** on **30 Apr 2005, 11:50 a.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Don Shepherd on 30 Apr 2005, 10:58 a.m.*

Hi Don,

My program looks for three unique digits, which I believe there are 47 solutions. I think there's only one solution for two unique digits. Not sure there's much insight in my program :) I'd be interested to see the 12CP version.. Not sure if the old 12C would have enough programming space to do it, but the 12cp has more space (?)

In looking over the command set for the 16C I just realized I could use RMD to isolate the right most digit. Would save a few steps.

If the base was switched from base ten to HEX or OCT, then the 16C would really shine. For example the square root of Hex number:

```
734877B53A10   is   ABCABC

and square root of the Oct Number:
14714571144    is   121212.
```

Using Hex or Oct, the bit operations would make the solution straight forward. I may try it just for fun.

Bill

Re: Math Challenge #8:- HP-12C Version

*Posted by **tony** on **30 Apr 2005, 10:50 p.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Bill (Smithville) on 30 Apr 2005, 11:50 a.m.*

Bill, this 57 liner seems to do the job on the 12C. It takes 20-25 seconds to fully analyse a candidate though, and the user interface is not that friendly<G>. Cheers, Tony

```
R0 hits number : 0.300040003 ->only 1,5,9 in square
R1 33333 ->
R2 10 (convenient constant)
R3 total of non-zero hits. If >3 we skip
R4 counter 0->10

Example:
33333 STO 1 R/S
after 20 seconds see 3
rcl 1->33,334  enter * ->1,111,155,556
rcl 0->0.500041000
imagine  123456789
showing 5 1s, 4 5s and 1 6.

R/S, see 3, RCL 1 ->33,335
R/S 20-25 seconds, see 33,336... 33,337... 33,338
another stop etc.....

Keystrokes      |Display       | Comments
[f][P/R]        |              |
[f]CLEAR[PRGM]  |00-           |
[EEX]           |01-        26 |
1               |02-         1 |
[STO]2          |03-    44   2 | R2=10
[CLx]           |04-        35 | top of loop1
[STO]0          |05-    44   0 | initialize hits
1               |06-         1 |
[STO][+]1       |07- 44 40   1 |
[RCL]1          |08-    45   1 |
[g][PSE]        |09-    43  31 | show current candidate
[ENTER]         |10-        36 |
[x]             |11-        20 |
[RCL]2          |12-    45   2 | top of loop2
[/]             |13-        10 |
[g][INTG]       |14-    43  25 | keep for recycling
[g][LSTx]       |15-    43  36 |
[g][FRAC]       |16-    43  24 |
[RCL]2          |17-    45   2 |
[x]             |18-        20 | digit found
[g][x=0]        |19-    43  35 | quick exit for 0
[g][GTO]04      |20- 43,33  04 | back to loop1
[CHS]           |21-        16 | otherwise add
[RCL]2          |22-    45   2 | 10^(-digit) to
[x<>y]          |23-        34 | the hits
[y^x]           |24-        21 |
[STO][+]0       |25- 44 40   0 |
[RDN]           |26-        33 | finish if remnant
[g][x=0]        |27-    43  35 | is zero. Otherwise
[g][GTO]30      |28- 43,33  30 | stay in loop2
[g][GTO]12      |29- 43,33  12 |
[STO]3          |30-    44   3 | initialise digit
[STO]4          |31-    44   4 | count & counter
[RCL]0          |32-    45   0 | analyse hits
[g][FRAC]       |33-    43  24 | top of loop3
```

```
[RCL]2          |34-    45   2 |
[x]             |35-        20 |
[g][INTG]       |36-    43  25 | augment counter
[STO][+]4       |37- 44 40   4 |
[g][LSTx]       |38-    43  36 | remnant
[x<>y]          |39-        34 |
[g][x=0]        |40-    43  35 | if no digits don't
[g][GTO]45      |41- 43,33  45 | augment count
1               |42-         1 |
[STO][+]3       |43- 44 40   3 |
[RDN]           |44-        33 |
[RDN]           |45-        33 |
[RCL]2          |46-    45   2 | if 10 digits
[RCL]4          |47-    45   4 | already found
[-]             |48-        30 | we are finished
[g][x=0]        |49-    43  35 |
[g][GTO]53      |50- 43,33  53 |
[RDN]           |51-        33 | otherwise
[g][GTO]33      |52- 43,33  33 | stay in loop3
3               |53-         3 |
[RCL]3          |54-    45   3 |
[g][x<=y]       |55-    43  34 | stop if 3 or less
[R/S]           |56-        31 | different digits
[g][GTO]04      |57- 43,33  04 | bottom of loop1
[f][P/R]        |              |
```

Re: Math Challenge #8:- HP-12C Version

*Posted by **Bill (Smithville, NJ)** on **2 May 2005, 1:12 p.m.**, in response to Re: Math Challenge #8:- HP-12C Version, posted by tony on 30 Apr 2005, 10:50 p.m.*

Hi Tony,

Great to see that the HP-12C can also do this. When I get a moment of time, I'll program it in. Thanks for posting it.

Bill

Re: Math Challenge #8:- HP-16C Version

*Posted by **Arnaud Amiel** on **2 May 2005, 9:05 a.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Bill (Smithville, NJ) on 29 Apr 2005, 1:14 p.m.*

I was also thinking that byte shifting is the way to go but I don't own a 16C so I used saturn assembly on the 49. It is then very easy to program.

Arnaud

Re: Math Challenge #8:- HP-16C Version

*Posted by **Bill (Smithville, NJ)** on **2 May 2005, 1:15 p.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Arnaud Amiel on 2 May 2005, 9:05 a.m.*

Hi Arnaud,

Well, byte shifting didn't come into as much use as I originaly thought - since it was base 10. Now if it was BCD....

I'm still planning on doing it in another base, such as HEX or OCT. Then byte shifting would really work.

Bill

Re: Math Challenge #8:- HP-16C Version

*Posted by **Arnaud Amiel** on **3 May 2005, 6:22 a.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Bill (Smithville, NJ) on 2 May 2005, 1:15 p.m.*

That is one good point of the saturn processor, I could code the whole thing in decimal mode.

Arnaud

Re: Math Challenge #8:- HP-16C Version

*Posted by **Valentin Albillo** on **4 May 2005, 4:41 a.m.**, in response to Re: Math Challenge #8:- HP-16C Version, posted by Bill (Smithville, NJ) on 29 Apr 2005, 1:14 p.m.*

Hi, Bill:

Bill posted:

"I have always enjoyed your Math Challenges but until now have never actually tried programming one."

Thanks for your interest and welcome to the club ! As you can see, the primary goal of my challenges is to provide fun, the kind of fun you get when you use your beloved calculator to try and find the answer to some interesting question, i.e.: to flex your 'calculateur extraordinaire' muscles, so to say. A secondary goal is to provide some didactic rewards in the process.

"I starting thinking that some sort of bit shifting could be used to solve it. So I got my HP-16C out, loaded it with new batteries and sit down last night with the manual to refresh my memory of how it operated and could it be used to solve this. I came up with the following 84 step program: [...] I'm sure the program could be optimized, since it's been 20 years since I have actually programmed with it."

Excellent attempt, indeed, and for an HP-16C no less ! There are so few programs out there for the 16C that I'm truly glad to see it used for this purpose. It is one thing to solve this challenge using and HP-71B or 42S, say, and quite another to do it with and HP-16C or 12C.

"Thanks to you Valentin, I have had a lot of fun getting re-aquainted with an old friend, the HP-16C."

You're welcome. And in more than one sense, hope to see you participating in my next challenges as well. You see, they aren't that difficult, they only seem to :-)

Best regards from V.

Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S

*Posted by **Marcus von Cube** on **4 May 2005, 12:38 p.m.***

This is a follow up on an archived message:

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv015.cgi?read=72151

I've ported my algorithm to my trusty TI-95 PROCALC, a machine I baught when they still were in the shops.

If you own the PC-Interface, you can compile the source with TIC95.EXE and load it into your calc.

The program presents a menu with 4 selections:

F1:GO Run the loop

F2:TST Run a single test on one number, displays the smallest digit.

F3:DIG Set number of different digits (2 or 3), 2 is default

F4:BEG Set beginning of loop, default is 33334.

For F3/F4, you have to enter the number and then press the key.

The most interesting result for me was that the TI-95 is noticably faster than my HP 42 (some 10 to 20% percieved).

The algorithm is essentially the same. The programming model in both calcs is comparable. TI's comparison statements work against registers (no longer the "t" register only). In many cases I had to store a value in a register on the TI where I could use the stack on the HP. Registers can be accessed by letters but this is just a shorthand notation to the corresponding numbers (RCL 000 is the same as RCL A).

The flags are used just the opposite way in my HP implementation and in this program: On the HP, I reset a flag when a corresponding digit is found while I set a flag on the TI. The reason is simple: the HP has a FC?C instruction that tests and clears a flag in one step. The feature is missing on the TI so I could use the more natural approach of setting the flags instead or clearing them.

Another missing feature on the TI is MOD: I had to use a more complicated formula to get the last digit.

Some statements can only be entered in so called unprotected (system) mode. These are SBA 226, a call to an assembly routine which does a VIEW (not available on the keyboard), and STB 00xx, which stores a byte in memory. The latter is used to clear the flags.

After entering the program you can speed it up with the ASM key: it replaces all (slow) label jumps by absolute jumps. INV ASM undoes the change.

Here is the program:

```
'
'  Find 10 digit squares with at most 2 or 3 different digits
'
       DFN CLR
       DFN F1:GO @GO          ' Start computation
       DFN F2:TST@TT          ' Test a single value
       DFN F3:DIG@DD          ' Set # of different digits (2, 3)
       DFN F4:BEG@BB          ' Set first guess
       2
       SBL DD                 ' Default: 2 different digits
       33334
       SBL BB                 ' First guess to square & test
       CLR
       HLT

LBL AA                        ' check a value against # of digits
       INC H                  ' some statistics
       SBA 226                ' display without delay
       STO A                  ' scratch register for value
       0
       STB 0028               ' system register: Flags 0-7
       STB 0029               ' system register: Flags 8-15
       STO C                  ' clear # of digits found so far
LBL A1
       (                      ' compute A MOD 10 and INT( A / 10 )
       RCL A
       -
       (
       /
       1
       0
       )
       INT
       STO A                  ' this is INT( A / 10 )
       *
       10
       )                      ' this is A MOD 10 (last digit of number)
       STO B                  ' use B for indirect flag setting
       INV TF IND B
         INC C                ' digit not yet encounterd, count it
       SF IND B               ' mark digit
       RCL C                  ' get count
       IF> D                  ' compare to maximum digits
         SF 00                ' too many digits
       TF 00                  ' tested number has failed
         RTN                  ' (0 encountered or too many digits)
       0                      ' all digits tested?
       IF= A
         RTN                  ' done
       GTL A1                 ' next digit

LBL CC                        ' get smallest valid digit for next guess
       RCL C                  ' # of digits found
       IF= D                  ' is the allowed maximum?
         GTL C1               ' find the smallest digit encounterd
       1                      ' not all allowed digits in last try
       RTN                    ' must return 1
LBL C1
       1                      ' digit index starts with 1
       STO B
```

```
LBL C2
       TF IND B                ' is digit in last tested number?
         RTN                  ' yes, must be the smallest
       INC B                  ' next digit
       RCL B                  ' get it (in case of return)
       GTL C2                 ' loop

LBL DD                         ' store # of allowed digits (key F3)
       STO D
       RTN

LBL BB                        ' store first number to square
       STO E                  ' compensate for INC E in main loop
       INV INC E
       0                      ' initialize counters and comparison values
       STO F                  ' first 4 digits
       STO G                  ' first 3 digits
       STO H                  ' total # of checks
       RTN

LBL GO                        ' main loop starts here (key F1)
       INC E                  ' next # to square
       99999                  ' check end of loop
       INV IF> E
         GTL ZZ               ' done
       (                      ' compute first 4 digits of squared #
       RCL E
       x^2
       /
       1000000
       )
       INT
       INV IF= F              ' same 4 digits as last try?
         SBL FF               ' no, get next guess based on first 4 digits
       RCL E                  ' get number again
       x^2                    ' and square it
       SBL AA                 ' digit test routine
       INV TF 00              ' check it test passed
         SBL PR               ' got it, show/print result
       GTL GO                 ' next guess

LBL FF                        ' find a new guess based on first 4 digits
       STO F
LBL F1
       RCL F                  ' 4 digits to test
       (                      ' check if first 3 digits have changed
       /
       10
       )
       INT
       INV IF= G
         SBL GG               ' check first 3 digits first
       RCL F                   ' get current guess
       SBL AA                 ' test it
       INV TF 00
         GTL F2               ' test passed: compute new guess for main loop
       INC F                  ' test failed, try next 4 digit number
       GTL F1
LBL F2                        ' compute a new guess
       (
       (
       RCL F                  ' first 4
       *
```

```
        10                      ' shifted 1 left
        +
        SBL CC                  ' add smallest possible next digit
        )
        *
        100000                  ' shift 5 left
        +
        11110                   ' offset which can be added safely
        )
        SQR                     ' the INT of the square root is next guess
        INT
        STO E                   ' store it
        INC E                    ' our guess was still too small
        RTN

LBL GG                          ' find a new guess based on first 3 digits
        STO G                   ' 3 digits to test
LBL G1
        RCL G                    ' get current guess
        SBL AA                  ' test it
        INV TF 00
          GTL G2                ' test passed: compute new guess for FF loop
        INC G                   ' test failed, try next 3 digit number
        GTL G1
LBL G2                          ' compute a new guess
        (
        RCL G                   ' first 3
        *
        10                      ' shifted 1 left
        +
        SBL CC                  ' add smallest possible next digit
        )
        STO F                   ' store it
        RTN

LBL PR                          ' print or show a valid result
        RCL E                   ' number to be squared
        PRT
        x~t                     ' save in t-Register
        RCL E
        x^2                     ' squared number in display
        PRT
        INV TF 74
          BRK                   ' halt if no printer connected
        RCL C                   ' number of different digits
        PRT
        (                       ' flag word
        RCB 0028
        +
        RCB 0029
        *
        256
        )
        HEX                     ' print it in HEX
        PRT
        DEC                     ' revert to normal display
        RTN

LBL TT                          ' test a single value (key F2)
        SBL AA                  ' call test routine
        0
        TF 00
        HLT                     ' stop and show 0 if test failed
        SBL CC                  ' show smallest digit or 1 if less then
        HLT                     ' the allowed number were found

LBL ZZ                          ' end of main loop
        RCL H                   ' show number of checks
        PRT
        HLT
END
```

Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S

*Posted by **Namir** on **4 May 2005, 12:51 p.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S, posted by Marcus von Cube on 4 May 2005, 12:38 p.m.*

Marcus .. you and I are soul brothers!!! I thought no one cared about the TI-95 machine. It is a cool programmable calculator and I am glad you are making it shine!

Namir

Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S

*Posted by **Marcus von Cube** on **4 May 2005, 2:08 p.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S, posted by Marcus von Cube on 4 May 2005, 12:38 p.m.*

Performance is better than I first thaught. After an hour, the calc had tested *all* 2 digit combinations (5142 total) and printed out the correct result 81619^2=6661661161.

Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S

*Posted by **HrastProgrammer** on **5 May 2005, 1:48 a.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S, posted by Marcus von Cube on 4 May 2005, 12:38 p.m.*

Hello Marcus,

Great work!

Could you, if you have some free time, of course, download my TI-95 emulator for Windows from http://www.geocities.com/hrastprogrammer, enter this program into the emulator and compare the result with the real calculator? As it is based on the real TI-95 ROMs I suppose the result must be the same but you never know ...

Best regards.

Re: Short & Sweet Math Challenges #8: TI-95 Emulator

Posted by ***Marcus von Cube*** *on* ***5 May 2005, 6:12 a.m.****, in response to Re: Short & Sweet Math Challenges #8: TI-95 beats HP 42S, posted by HrastProgrammer on 5 May 2005, 1:48 a.m.*

Hi Hrastprogrammer!

I first found TI95E in September 2004. I checked the download and it's still the most recent version. (I'd like to see the PC interface emulated, a cartridge add/remove dialog, ...)

The program yields the exact same results as on the real machine, except the time (in full speed mode): The complete 2 digit run was a matter of a minute!

Steps to transfer the program:

1. On the real machine, store it in the cartridge

2. Use the PC interface and program PCA to save the cartrige to a file.

3. Transfer the file to the emulator directory and use my program ti95cart to incorporate the cartrige file in the memory image.

The program can than be started from the emulated RAM.

If you drop me a mail, i'll respond with my files.

Marcus

P.S.: The emulator runs fine under OS/2 with Odin installed! I still prefer it over Windows on my main system.

Re: Short & Sweet Math Challenges #8: TI-95 Emulator

Posted by ***HrastProgrammer*** *on* ***5 May 2005, 9:47 a.m.****, in response to Re: Short & Sweet Math Challenges #8: TI-95 Emulator, posted by Marcus von Cube on 5 May 2005, 6:12 a.m.*

I first found TI95E in September 2004. I checked the download and it's still the most recent version. (I'd like to see the PC interface emulated, a cartridge add/remove dialog, ...)

I would like to see these things implemented, as well :-) But I don't have any info about the interface and I don't have any cartridge to test module loading :-(

You have my e-mail address on my homepage so you are welcome to send your files ...

Best regards.

Re: Short & Sweet Math Challenges #8: TI-95 Emulator

*Posted by **Marcus von Cube** on **6 May 2005, 2:36 a.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 Emulator, posted by HrastProgrammer on 5 May 2005, 9:47 a.m.*

Hi HrastProgrammer,

you can find a lot of technical info on the following ftp server:

ftp://ftp.whtech.com/Hexbus-%20CC40+TI74/

You'll need the free Scansoft PaperPort Viewer to read the ".max" files.

I've put my files (and some of TI's) here: http://www.mvcsys.de/download/ti95.zip

(ti95card.exe is an untested DOS compile but the source is included)

Re: Short & Sweet Math Challenges #8: TI-95 Emulator

*Posted by **HrastProgrammer** on **6 May 2005, 9:28 a.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 Emulator, posted by Marcus von Cube on 6 May 2005, 2:36 a.m.*

Thank you for the files, especially for MATH and STAT cartridges.

I modified my TI-95 emulator so now cartridges can be loaded, saved and cleared ... You can download the new version from my homepage http://www.geocities.com/hrastprogrammer.

Best regards.

Amazing

*Posted by **Gene** on **6 May 2005, 9:54 a.m.**, in response to Re: Short & Sweet Math Challenges #8: TI-95 Emulator, posted by HrastProgrammer on 6 May 2005, 9:28 a.m.*

I can't believe you cranked that out so fast. Thanks!

Re: Amazing

*Posted by **HrastProgrammer** on **6 May 2005, 10:24 a.m.**, in response to Amazing, posted by Gene on 6 May 2005, 9:54 a.m.*

I can't believe you cranked that out so fast. Thanks!

I had to take a break while working on Voyager emulators :-)

You tease! :-) (n/t)

*Posted by **Gene** on **6 May 2005, 10:52 a.m.**, in response to Re: Amazing, posted by HrastProgrammer on 6 May 2005, 10:24 a.m.*

I said nothing was in here. :-)

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Arnaud Amiel** on **13 May 2005, 1:46 a.m.***

This is a follow up to an archived message.

Ihad posted a Saturn Assembly solution to this challenge but felt it could do with some tiyding up. So I did and gained a few seconds. Then I realised that most of the time was spent squaring the numbers. Knowing that the sum of odd numbers is the list of square numbers, I could change the squaring into a mere couple of additions. This code now runs much faster.

It takes 5.3s on a 49g+ and 13s on a 48GX. I am not talking about emulated speed here but real speed. I still need to port it to the 28S. The 71 has a Saturn but I don't have one and don't know what is its support for assembly...

I can send you the compiled program if you email me.

Arnaud

```
SAVE SETDEC
LA 1111088889000
C=0.M LC 66665000 B=C.M
LC 33333 RSTK=C

*NextNumber
C=RSTK C+1.A
SKIPNC { P=0 SETHEX LOADRPL }
RSTK=C
B+1.M B+1.M A+B.M C=A.M     %This is were the square is calculated

P=5 A=0.XS B=0.XS D=0.XS
*TestNumber
P+1 GOC GoodNumber
CSR.W
?C=0.XS GOYES NextNumber
?A#0.XS { A=C.XS GOTO TestNumber }
?A=C.XS GOYES TestNumber
?B#0.XS { B=C.XS GOTO TestNumber }
?B=C.XS GOYES TestNumber
?D#0.XS { D=C.XS GOTO TestNumber }
?D=C.XS GOYES TestNumber
GOTO NextNumber

*GoodNumber
R1=A.M
C=B.M R2=C.M
ASL.M ASL.M
LA 009
A=0.S
GOSBVL =PUSH%
SAVE
SETDEC
A=R1.M
C=R2.M B=C.M
GOTO NextNumber
@
```

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Namir** on **13 May 2005, 3:34 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Arnaud Amiel on 13 May 2005, 1:46 a.m.*

Arnaud,

All these challenges are not part of some post graduate homework you have been give (<big smile>). Was just wondering!

Namir

(Basking in the sunny and sometimes cloudy southern France ... for a few days more)

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Arnaud Amiel** on **13 May 2005, 4:24 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Namir on 13 May 2005, 3:34 a.m.*

Unfortunately, last time I was given some postgraduate work to do was about 5 years ago, what is more since then I have been basking in the British rain instead of the warm south of France ever since...

Arnaud

Re: Short & Sweet Math Challenges #8: Squares !

*Posted by **Namir** on **13 May 2005, 6:16 a.m.**, in response to Re: Short & Sweet Math Challenges #8: Squares !, posted by Arnaud Amiel on 13 May 2005, 4:24 a.m.*

Arnaud,

You are missing the beauty and magic of southern France. They predictewd clouds today ... well not for long. It's sunny and beautiful at noon today!! It's hard to stay inside and play with the internet and HP emulators. I am surrendering to the temptation of this nice weather.

Namir

Short & Sweet Math Challenge #9: Divisibility !

Posted by ***Valentin Albillo*** *on* ***16 May 2005, 9:31 a.m.***

Hi, all:

("Di-vi-si-bi-li-ty" ... far too many 'i's for just one single word, reminds me of the infamous song *"Miss Lilly Higgins sings shimmy in Mississippi's spring" ...*

Here's the brand new **S&SMC#9** I've just concocted for your HP-programming pleasure, let's hope it gathers as much interest among forum contributors as the previous one did (still getting answers long after it was duly archived). Just like it, this one can also be addressed using just about any HP calculator, combined with a generous helping of ingenuity on the part of the user. Enough intro, let's go right to the

Challenge:

- **(1)**. Find a 10-digit integer number which consists of the digits 0-9 *without repetition* such that taking its L (from 1 to 10) leftmost digits the resulting number is *exactly divisible* by L. For instance, the number 32165 would be a solution, as it has no repeating digits and
- `           3 is exactly divisible by 1 =>  3/1 = 3`
- `          32 is exactly divisible by 2 =>  32/2 = 16`
- `         321 is exactly divisible by 3 =>  321/3 = 107`
- `        3216 is exactly divisible by 4 =>  3216/4 = 804`
- `       32165 is exactly divisible by 5 =>  32165/5 = 6433`

  except for the unfortunate fact that it's got only 5 digits while we're after a 10-digit number. **The solution is unique**. As usual, you're expected to produce a program for your chosen calculator(s) that upon running will find the only solution *and* make sure there are no others.

  If you succeed, you may want to extend your quest to also:

- **(2)**. Find out just *how many* solutions there are for N-digit numbers, with N ranging from 1 to 10: for N=10 there's just one solution, but what about N=4 or N=9 ? Numbers beginning with a 0 are to be disregarded, *including "0" itself* (i.e.: "0" is *not* to be considered a valid solution for the case N=1). Your program must output just the number of solutions for each N, displaying the solutions themselves is not required.

- **(3)**. Same as **(2)** above, but *repeated* digits *are* allowed, the divisibility condition and not beginning with '0' being the only requirements.

As stated, you must produce one or several programs for your HP calculator(s) that will compute what's required, the shorter and faster the better. Giving *just* the solutions (but no program) or providing programs for machines other than (preferably HP) *calculators* is to be considered as blatant disrespect to the stated rules and/or a clear statement of utter inability to comply within these terms.

Two or three days from now I'll post my solution (plus comments), which is essentially a rather simple, unoptimized 4-line program (plus variations) for a bare-bones HP-71B that takes less than 2 minutes to find the only solution to **(1)**, less than half an hour to count all 706 solutions to **(2)**, and less than 4 hours to count all 11,453 solutions to **(3)** (in a physical HP-71B, that is; emulated times for Emu71 @ 2.4 Ghz are 1 second, 8 seconds, and 10 minutes, respectively).

These are not the best times possible in a 71B, of course. In order for you to test your programs, results for **(2)** and **(3)** when N=4 are 168 and 375 solutions, respectively.

Let's see your efforts. If you can't solve this challenge for N up to 10, by all means do try with a lower limit for N, say 5, but do try all the same. Good luck and happy programming, and above all, enjoy !

Best regards from V.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **GWB** on **16 May 2005, 9:46 p.m.**, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Hi Valentin,

So far I have prefered to do it by hand. I just had to make a table of the thirty or so possible solutions according to the first four digits. After eliminating a few that wouldn't fit, I found the solution. (I think it would take longer, at least for me, to get to the solution by writing a program). Very interesting problem, though.

In order to respect your rules, I won't give the solution here. I will just say the first two digits of the solution are the number of an HP calculator, so are the third and fourth digits, the fourth and fifth digits and the eigth and ninth digits, right? I'm looking forward for your and other people's solution.

Regards,

Gerson.

Just in case, here is my table:

```
1236 1296 1472 1476 1832 1836 1892
3216 3276 3692 3698 3812 3816 3872
3876 7236 7296 7412 7416 7832 7836
7892 9216 9276 9632 9812 9816 9872
9876
```

As everybody have seen, odd-position digits have to be odd and even-position digits have to be even. Also, the 5th digits is 5 and the 10th digit is 0. So, beginning with 12365 and trying the remaining even digits 4 and 8, both 123656 and 123658 are not divisible by 6, so it is discarded. After trying 12 more numbers, the solution is eventually found: 3816547290. This scheme is hard, if impossible, to put in an algorithmic form though.

Edited: 18 May 2005, 6:49 p.m.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Eamonn** on **16 May 2005, 10:20 p.m.**, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Hi Valentin,

Here is a HP-32S solution for the first part of the challenge. It finds the single solution, 3816547290. Running time is 132 seconds.

To use the program, enter GTO P, then R/S. The program stops to display any solutions it finds. It finishes with a branch to a non-existent label. This makes it more obvious to tell when the program has stopped vs. when it is displaying a solution.

I had to resort to some optimizations and trickery to get around the restrictions of the HP-32S. In doing so, I made use of the following observations:

- Since a 2-digit number must be divisible by 2, it must therefore end in an even digit. By the same token, a four, six, eight or ten digit number must also end in an even number.
- For a ten digit number to be divisible by 10, it must end in a zero.
- Since all the even-numbered digits of the solution(s) are even, therefore all the odd-numbered digits must be odd.

Two outer loops in the program generate all the two digit numbers that have the first digit odd and the second digit a non-zero even number. A recursive subroutine generates more digits, making sure that no digits are re-used and testing each time for divisibility by the number of digits in the number. If we ever get a 9-digit number, then it automatically means that we have found a ten digit solution, since the digit zero was not used to form the 9-digit number.

The program should be easily modifiable to solve the second part of the challenge - I'll post it if/when I get some time to complete it. I'll need to think a bit more of how to fit the third part of the challenge into the HP-32S. Restricting the solutions to eight digits may be necessary because the seven levels of stack depth on this device may not be enough.

Thanks for the challenge.

Eamonn.

```
LBL P
0
STO Y     ; Flags Variable
STO W     ; Count of numbers found
3
STO i     ; Divisor
10
STO X     ; Handy Constant
1.00902   ; For A = 1 to 9 Step 2
STO A
[23 Bytes Chksum = 43DE]

LBL A
RCL A
IP
10^X
STO+ Y    ; Flags[A] = 1
2.00802   ; For B = 2 to 8 Step 2
```

```
STO B
[18.5 Bytes Chksum = A74D]

LBL B
RCL B
IP
10^X
STO+ Y    ; Flags[B] = 1
RCL A
RCL* X
RCL+ B
IP
STO M     ; M = A * 10 + B
XEQ D     ; call recursive function
RCL B
IP
10^X
STO- Y    ; Flags[B] = 0
ISG B
GTO B     ; Next B
RCL A
IP
10^X
STO- Y    ; Flags[A] = 0
ISG A
GTO A     ; Next A
RCL Z     ; display the count of numbers found
GTO Z     ; Program finishes on an error - Z is a non-existent label
[37.5 Bytes Chksum = EB63]

LBL D     ; Recursive function
2.00802
STO (i)   ; For odd digits, test only odd numbers
RCL i
2
/
FP
X=0?
GTO L
1.00902   ; Even digit - so test even numbers
STO (i)
[32.5 Bytes Chksum = 6C06]

LBL L
RCL Y     ; Y contains the flags
RCL (i)   ; Code to test flag (i)
IP
10^X
/
IP
RCL/ X    ; X = 10
FP
X<>0?
GTO E     ; if Flags[LCV[i]] <> 0 then done
RCL (i)
IP
10^X
STO+ Y    ; Flags[LCV[i]] = 1
RCL M
RCL* X
RCL+ (i)
IP
STO M     ; M = M * 10 + LCV[i]
```

```
RCL i
/
FP
X<>0?
GTO F     ; If M is not divisible by i then goto f
9
RCL i
X=Y?
GTO G     ; if i is already 9 then answer is found
1
STO+ i
XEQ D     ; Else increment i and test for next digit
1
STO- i
GTO F
[52.5 Bytes Chksum = 4385]

LBL G     ; We are here if 9-digit number found
RCL M
RCL *X    ; Display M*10
STOP
STO V     ; Store the result
1
STO+ W    ; Increment count
[10.5 Bytes Chksum = 7F10]

LBL F     ; Some Cleanup
RCL M
RCL/ X
IP
STO M     ; M = M / 10
RCL (i)
IP
10^X
STO- Y    ; Flags[LCV[i]] = 0
[13.5 Bytes Chksum = 9D81]

LBL E
ISG (i)   ; Next LCV[i]
GTO L
RTN
[6 Bytes Chksum = 4EC6]
```

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Arnaud Amiel** on **18 May 2005, 4:14 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Eamonn on 16 May 2005, 10:20 p.m.*

I haven't have time to give it a try yet but an other simplification is that the 5th number has to be 5 or 0 and as the 10th number has to be 0, the 5th is 5.

I didn't go any further than that and may not before the weekend.

Arnaud

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **J-F Garnier** on **18 May 2005, 7:54 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Arnaud Amiel on 18 May 2005, 4:14 a.m.*

Going one step further in this approch to solve Valentin's first problem:

The 4-digit number being divisible by 4, and the 3rd digit being {1,3,7,9}, the 4th digit is either 2 or 6.

The 8-digit number being divisible by 8, the 7th digit being {1,3,7,9}, and the 6th being even, the only combinations of digits 7th and 8th are: 16, 32, 72, 96, so the 8th digit is either 2 or 6.

As digits 4th and 8th are either 2-6 or 6-2, the two other even digits 2nd and 6th are either 4-8 or 8-4.

Finally, there are 48 remaining candidates: 24 combinations for the 4 odd digits 1st, 3rd, 7th and 9th, digit 7th fixing digit 8th and in turn digit 4th, and 2 combinations for digits 2nd and 6th.

Below is the (probably) fastest HP-71B program to find the solution:

```
 10 DATA 143658729,143258967,147658329,147258963,149658327,149658723
 20 DATA 341658729,341258967,347258169,347258961,349258167,349658721
 30 DATA 741658329,741258963,743258169,743258961,749258163,749658321
 40 DATA 941658327,941658723,943258167,943658721,947258163,947658321
 50 DATA 183654729,183254967,187654329,187254963,189654327,189654723
 60 DATA 381654729,381254967,387254169,387254961,389254167,389654721
 70 DATA 781654329,781254963,783254169,783254961,789254163,789654321
 80 DATA 981654327,981654723,983254167,983654721,987254163,987654321

 90 FOR I=1 TO 48
100   READ A
110   IF MOD(A,9)=0 THEN
120     X=A
130     X=X DIV 100
140     IF MOD(X,7)=0 THEN
150       X=X DIV 10
160       IF MOD(X,6)=0 THEN
170         X=X DIV 1000
180         IF MOD(X,3)=0 THEN
190           DISP A*10
200         END IF
210       END IF
220     END IF
230   END IF
240 NEXT I
```

(Using the IF-ENDIF structure, and listed with PBLIST command from JPCROM)

J-F

Edited: 18 May 2005, 8:20 a.m.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Marcus von Cube** on **18 May 2005, 8:36 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by J-F Garnier on 18 May 2005, 7:54 a.m.*

The probably fastest program, following this approach, is to just enter the solution and print it out ;-)

The problem is that all these simplifications will only work for 10 digit numbers. If you test for less digits, you'll have to test many more candidates.

My idea is, not build the numbers with rules like the ones mentioned until now and check for divisibility, but to go the other way round:

1) Start with the left most digit, setting it to 1.

2) Find a starting point for the nth digit by adding a 0 to the right of the number found so far, dividing the result by n (integer divide), adding 1 and multiplying again by n. This gives the first number to test and should preserve the digits to the left.

3) If the number satisfies the condition(s) (i. e. has no duplicate digits for problems (1) or (2)), increase the divisor n and goto step 2. The algorithm has found a solution, if n is big enough. It stops, if we get an overflow with n=1 (the left most digit was already 9).

4) Try another number by adding the divisor n. If there is no overflow to the digit to the left, repeat step 3.

5) If adding n to the last digit would cause an overflow, we must decrease n, drop the last digit and go back to step 4.

I haven't implemented the algorithm so far but I feel that can be done on a 41 or 42S or even a smaller device (a 16C in integer mode?)

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Eamonn** on **19 May 2005, 7:06 p.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Eamonn on 16 May 2005, 10:20 p.m.*

I finally had time to look into a solution for parts 2 and 3 of S&SMC9. Initially I was hoping to write a program for the HP-32S. I could easily modify my previous program to count the number of solutions for part 2 and part 3 that there are for a single n-digit number. This program could then be run for each value of n.

To count the number of solutions for a n-digit number, the program generates all the solutions for n-1, n-2, ... digits etc. To calculate the number of solutions for 3 digits, the program calculates all the 2-digit and 1-digit solutions. To calculate the number of solutions for 4 digits, all the work that was previously done to count the number of 3-digit solutions must be re-done. Not very efficient.

Ideally, the program would just count the number of solutions for n=10, and store the number of solutions found for n=9, 8, etc along the way. However, the HP-32S doesn't have enough memory to store both the program and the counts of the solutions.

I decided to go with a calculator with more memory. Unfortunately, I don't have in my possesion a HP calculator with more memory than the HP-32S. So, I went ahead and wrote a program in Basic for the Sharp PC-1262. It's not a HP, and some would say that it's more a pocket computer than a calculator, but the latter could also be said of the HP-71B.

Anyway, here is my solution for parts (2) and (3) of the challenge. For part (2), run the program as-is. For part (3), change the first statement on line 10 to V=0. The main optimization it to only consider even-valued numbers for the even digit positions, the values 0 and 5 for the fifth digit position and the value 0 for the tenth digit position. When the program finishes, the number of solutions for a n-digit number can be found by looking at C(n).

```
10 V = 1: I = 2: DIM C(10): DIM L(10): DIM Z(10): DIM F(10): FOR A = 1 TO 10:
READ Z(A):NEXT A
20 FOR A = 1 TO 9: N=A: F(A) = V: C(1) = C(1) + 1: GOSUB 30: F(A) = 0: NEXT
A: END
30 L(I) = 0: N = N * 10
35 IF F(L(I)) = 1 THEN 70
40 Y = N / I: IF Y <> INT(Y) THEN 70
50 C(I) = C(I) + 1: IF I = 10 THEN 70
60 F(L(I)) = V: I = I + 1: GOSUB 30: I = I - 1: F(L(I)) = 0
70 L(I) = L(I) + Z(I): IF L(I) < 10 LET N = N + Z(I): GOTO 35
80 N = INT(N/10): RETURN
90 DATA 1, 2, 1, 2, 5, 2, 1, 2, 1, 10
```

It takes about 15 minutes to find the following solutions for part (2)

```
#digits  #solutions
 1 ----    9
 2 ----   41
 3 ----  115
 4 ----  168
 5 ----  220
 6 ----   88
 7 ----   51
 8 ----    9
 9 ----    4
10 ----    1
```

It takes about 4 hours to find the following solutions for part (3)

```
#digits  #solutions
 1 ----     9
 2 ----    45
 3 ----   150
 4 ----   375
 5 ----   750
 6 ----  1200
 7 ----  1713
 8 ----  2227
 9 ----  2492
10 ----  2492
```

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Valentin Albillo** on **20 May 2005, 4:56 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Eamonn on 19 May 2005, 7:06 p.m.*

Hi, Eamonn !

My most sincere congratulations, your answers to S&SMC#9, all three variants, are absolutely *correct*, and with very good timings as well, for the models used. Which is more, you solved (1) in such a RAM-challegend calculator as the HP32S and (2)-(3) in a BASIC dialect that doesn't allow for recursion, which is also quite bold on your part.

The SHARP PC-1262 you used is a really, really, really wonderful machine, absolutely comparable if not superior to anything HP had to offer at the time, and even today. Physically, it was very similar (an upgrade, actually) to the SHARP PC-1260, this is one of mine (actual size is more or less like an HP-15C):

Just a small comment on your BASIC version. Lines such as

```
    DIM C(10): DIM L(10): DIM Z(10): DIM F(10)
```

can be entered instead as

```
    DIM C(10),L(10),Z(10),F(10)
```

which if more readable, as well as shorter and faster. Also, lines such as

```
    N=A: F(A) = V: C(1) = C(1) + 1
```

can be entered instead as

```
    N=A, F(A) = V, C(1) = C(1) + 1
```

which is faster, and probably more readable as well.

Since it seems no more people are taking the challenge (too difficult, perhaps ?) I'll give my solutions soon.

Thank you very much for your continued interest *AND* superb contributions and

Best regards from V.

Edited: 20 May 2005, 5:11 a.m.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Eamonn** on **21 May 2005, 1:39 p.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 20 May 2005, 4:56 a.m.*

Hi Valentin,

Thanks for the feedback and the Basic programming tips - I only recently purchased the PC-1262, without manual, and it is indeed a very nice machine with really great capabilities.

This was a good challenge on which to try out Sharp Basic programming. It certainly was a lot faster to develop on the Sharp than on the HP, although that may be partly because I had already done most of the work already on the HP.

One thing I like about your challenges is that they are very well designed. It's always interesting to see the approaches everyone takes to solving them and to see just what can be done on a vintage machine with limited resources. Looking forward to the next challenge.

Best Regards,

Eamonn.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Valentin Albillo** on **23 May 2005, 9:18 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Eamonn on 21 May 2005, 1:39 p.m.*

Hi, Eamonn:

Eamonn posted:

"I only recently purchased the PC-1262, without manual, and it is indeed a very nice machine with really great capabilities."

Yes, indeed, one of the very best in its class. As for the manual, its BASIC syntax and capabilities are very much like other SHARP models, so you can use their manuals profitably with it.

"This was a good challenge on which to try out Sharp Basic programming. It certainly was a lot faster to develop on the Sharp than on the HP, although that may be partly because I had already done most of the work already on the HP."

Among the many HP and SHARP models I've tried, the fastest one for program development was the SHARP PC-1350 or -1360, thanks to its large display (4 lines x 24 characters). You can develop complicated programs starting from absolute scratch without ever committing a single line of code to paper.

"One thing I like about your challenges is that they are very well designed. It's always interesting to see the approaches everyone takes to solving them and to see just what can be done on a vintage machine with limited resources. Looking forward to the next challenge."

Thank you very much for your interest, I'm glad you enjoyed the challenge, I was also quite happy with your successful efforts to solve it. Next challenge in a couple of weeks or so.

Best regards from V.

Negligible suggestion

Posted by ***Andrés C. Rodríguez (Argentina)*** *on* ***17 May 2005, 10:46 p.m.****, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Valentin: while my current work schedule prevents me from accepting your (once again) excellent challenges (at least when there is no weekend included in the allowed time interval), I would just like to point that (despite the manner the song title appears in many Internet sites) I recall it as "Miss Lilly Higgins sings shimmy in Mississippi Springs"; a place in the real USA, indeed.

You may also like the sequel "Doctor Bob Gordon shops hot dogs from Boston"; also from Les Luthiers, of course.

Best regards, and thank you for the challenges!

Re: Negligible suggestion

*Posted by **Valentin Albillo** on **18 May 2005, 4:47 a.m.**, in response to Negligible suggestion, posted by Andrés C. Rodríguez (Argentina) on 17 May 2005, 10:46 p.m.*

Hi, Andrés ! :-)

Andrés posted:

"... there is no weekend included in the allowed time interval"

You have a point there. Certainly, posting the S&SMCs on Fridays, say, would allow interested people to tackle them at leisure during the weekend, when there's more free time and the mind is (relatively) free from the week's work chores. So I'll proceed that way from now on, thanks a lot for pointing this out.

"I would just like to point that (despite the manner the song title appears in many Internet sites) I recall it as "Miss Lilly Higgins sings shimmy in Mississippi Springs"; a place in the real USA, indeed."

Regrettably, I don't have the CD at hand to look the exact name of the song appearing in there, but I'm almost sure it's like I posted. I think your version makes more sense, at least to me, but if they called their song that way, so be it.

"You may also like the sequel "Doctor Bob Gordon shops hot dogs from Boston"; also from Les Luthiers, of course."

And, which of course, is a "foxtrot" ! (what else ?) :-) Not to mention their *"Papa Garland had a hat and a jazz band and a mat and a black fat cat"*, which is also priceless.

I do have their full discography and 'videography' in CD and DVD and if you're ready for an extra thrill, do watch any of their shows in DVD *with the English subtitles on!!*. Seeing the tremendous, heroic efforts to try and translate their extremely humoristic word plays and innuendos into English is fun beyond belief, specially as the translator mostly succeeds, against all odds !. Certainly, "Les Luthiers" are a real, real asset to your marvelous people and country.

Again, thanks for your interest and

Best regards from V.

Teen's solution (Re: Short & Sweet Math Challenge #9: Divisibility !)

Posted by ***GWB*** *on* ***18 May 2005, 8:53 p.m.****, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Gentlemen,

I've just found this (hand) solution by a 13-year old schoolboy from New Zealand:

http://www.nrich.maths.org.uk/public/viewer.php?obj_id=796&part=solution&refpage=viewer.php

Regards,

Gerson (43)

Edited: 18 May 2005, 8:59 p.m.

The Basket Case

*Posted by **whuy** on **19 May 2005, 7:52 a.m.**, in response to Teen's solution (Re: Short & Sweet Math Challenge #9: Divisibility !), posted by GWB on 18 May 2005, 8:53 p.m.*

That same site has a nice problem too, that can serve as another SSMC: The Basket Case
http://www.nrich.maths.org.uk/public/viewer.php?obj_id=1137&part=index&refpage=monthindex.php

A girl buys 4 items, but mistakenly multiplies their prices instead of adding them up. She arrives at a total of $7.11, which turns out to be the correct total. What were the respective prices of the 4 items?

I've been able to solve it using a 48/49 program.. no doubt Valentin can write it in 2 or 3 lines on his 71B ;-)

Cheers, Werner

Re: The Basket Case

Posted by ***Valentin Albillo*** *on* ***19 May 2005, 11:34 a.m.****, in response to The Basket Case, posted by whuy on 19 May 2005, 7:52 a.m.*

Hi, Werner:

Werner posted:

"A girl buys 4 items, but mistakenly multiplies their prices instead of adding them up. She arrives at a total of $7.11, which turns out to be the correct total. What were the respective prices of the 4 items? [...] I've been able to solve it using a 48/49 program.. no doubt Valentin can write it in 2 or 3 lines on his 71B ;-) "

With this kind of 'calculator challenges' one has to strive for a proper compromise between these two extremes:

- on the one hand, one does think very little about the problem and opts instead for a pure brute force approach. The resulting program is very short but extremely 'brute' and runs for ages before finding any solutions. No good.
- on the other hand, one does think a whole lot about the problem, and after a long mental effort one manages to reduce the search to much fewer cases, so the resulting program incorporates all this hardly-won knowledge, and does very little search. Why, the *user* did the search for the program, so it might run much faster but at the expense of lots of user's time and effort. No good, either.

I advocate that the proper compromise between these two extremes is to give the problem a little think, but not too much, just the obvious elimination of provably impossible cases and the like, then write a program that will actually do the search, as it was supposed to do to begin with.

With respect to your problem, and accordingly to this challenge-solving philosophy, I considered the problem for a few minutes, and extracted the following fast guidelines:

- The problem deals with prices in cents, so all numbers appearing here are in multiples of 0.01. It's best to get rid of those fixed-point decimals and use integers instead. This means the program will search for numbers adding up to 711 and whose product is 711*100*100*100. Once a solution is found, we'll simply scale down its numbers by 100.
- If integer numbers must have a product equal to 711000000, at least one of them must be equal to this product's largest prime divisor or a multiple thereof. For this number (711), a very quick test reveals that its largest prime factor is 79. So one of the (scaled-to-integer) prices, say A, must be either 79 or a multiple of 79.
- The remaining prices, B,C, and D, are bound by simple limits regarding to their sum and/or their divisibility of the product, which cost nothing to take into account.

I then purposely refrained from spending any more *manual* effort on my part, and wrote instead a short, simple program incorporating those quick discoveries, namely this 5-liner for the bare-bones HP-71B, which took me just 5 minutes to write and test:

```
    1  K=711 @ D=79 @ M=100 @ R=K*M^3 @ S=R/D @ FOR A=D TO K-D STEP D
    2  FOR B=1 TO (K-A)/3 @ IF MOD(S,B) THEN 5 ELSE T=A+B @ F=A*B
    3  FOR C=B TO (K-T)/2 @ IF F*C*(K-T-C)=R THEN DISP A/M;B/M;C/M;(K-T-C)/M
@ END
```

```
    4  NEXT C
    5  NEXT B @ NEXT A @ DISP "Not found"
```
Upon running, it displays:
```
    >RUN

        3.16  1.2  1.25  1.5
```
which is a correct solution as both the sum and the product of these quantities do equal 7.11.

The running time is some 30 seconds under Emu71 @ 1.4 Ghz, and thus should be some 10-15 minutes in a physical 71B.

Best regards from V.

P.S.: Should you *insist* in your *"no doubt Valentin can write it in 2 or 3 lines on his 71B ;-)"* remark, this version
```
    1  K=711 @ D=79 @ M=100 @ R=K*M^3 @ S=R/D @ FOR A=D TO K-D STEP D @ FOR
B=1 TO (K-A)/3
    2  T=A+B @ F=A*B @ FOR C=B TO (K-T)/2 @ IF F*C*(K-T-C)=R THEN DISP
A/M;B/M;C/M;(K-T-C)/M @ END
    3  NEXT C @ NEXT B @ NEXT A @ DISP "Not found"
```
does it in just 3 lines by omitting a single divisibility criterium, but it takes 3-4 times as long to find the solution. Compromise, compromise !

Edited to remove a couple of typos

Edited: 19 May 2005, 12:19 p.m.

Re: The Basket Case

*Posted by **whuy** on **19 May 2005, 3:15 p.m.**, in response to Re: The Basket Case, posted by Valentin Albillo on 19 May 2005, 11:34 a.m.*

Hi, Valentin! It is the only solution, as well. I took a slightly different (worse..) approach ,but managed to find it in 'reasonable time' as well. Hope you didn't spoil the fun for everyone else, posting an excellent solution so quickly. The one thing that aggravates me about the programs you post for the 71B is the fact that you quit a loop early in just about each one of them - and how easy it is in the 71B. And what a mess it is on the 48/49..

Cheers, Werner

Re: The Basket Case

*Posted by **Valentin Albillo** on **20 May 2005, 4:43 a.m.**, in response to Re: The Basket Case, posted by whuy on 19 May 2005, 3:15 p.m.*

Hi, Werner ! :

Werner posted:

```
    "I took a slightly different (worse..) approach ,but managed to find
     it in 'reasonable time' as well."
```

Why don't you post it ? I always have a lot of fun looking at those unfathomable RPL listings ... ;-)

```
    "Hope you didn't spoil the fun for everyone else, posting an excellent
     solution so quickly."
```

Well, thanks for your kind comment but you were posting a *different* challenge of yours in *my* original challenge's thread, instead of posting anything related to it, and that's pretty unpolite to begin with. If you don't want your new challenge "spoiled", go and create your own thread ! :-)

```
    "The one thing that aggravates me about the programs you post for the
     71B is the fact that you quit a loop early in just about each one of
them -
     and how easy it is in the 71B. And what a mess it is on the 48/49.."
```

Everything is a mess in RPL. Where you had the simplicity and sheer joy of keystroke programming, you've got a "structured", high-level language that will force you to do things *its* way instead of allowing you to do things *your* way, that you'll never entirely master and that will have you spend inordinate amounts of time to try and remember how to do the simplest things. Where you had the simplicity and extreme readability of BASIC, you've got an unfathomable bunch on nearly incomprehensible, ghastly-looking commands and statements that no one, not even the original programmer, can understand later except by careful and painful analysis, if at all ...

And yes, you'll be amazed to see what can be done on a 71B in just 3-4 lines of code, and how easy and natural it all is. If in doubt, have a look at my "Baker's Dozen" articles in Datafile, specially the second one (Vol. 2, alas not yet online, you'll need to get the corresponding Datafile issue).

Thanks for your interesting challenge and comments, and

Best regards from V.

Bad Languages (OT)

*Posted by **Thomas Okken** on **20 May 2005, 7:06 a.m.**, in response to Re: The Basket Case, posted by Valentin Albillo on 20 May 2005, 4:43 a.m.*

**Everything* is a mess in RPL.*

Hear, hear!
Once we're done hunting down and roasting over a slow fire the persons responsible for RPL, let's go after the idiot who decided that Java would not have **goto**. ;-)

Long ago, I did a bit of PostScript programming. The result was almost as psychedelic as Lisp. If programming is all about the satisfaction of getting complex and difficult work done, these weird languages are a godsend, because they make *everything* difficult...

- Thomas

Re: Bad Languages (OT)

*Posted by **Valentin Albillo** on **20 May 2005, 7:42 a.m.**, in response to Bad Languages (OT), posted by Thomas Okken on 20 May 2005, 7:06 a.m.*

Hi, Thomas:

Thomas posted:

"Once we're done hunting down and roasting over a slow fire the persons responsible for RPL, let's go after the idiot who decided that Java would not have goto. ;-)"

Agreed. Look for some Wickes guy for the first abomination, don't know about the Java culprits.

*"If programming is all about the satisfaction of getting complex and difficult work done, these weird languages are a godsend, because they make *everything* difficult..."*

Agreed. And the saddest thing of all is that, in the end, all those fancy, goto-forbidding languages are ultimately translated into machine-language programs, *where statistics say that 30-35% of all instructions are branching instructions*, aka **goto**'s !

I'd like to see all those nerdy goto-haters try and write a compiler which does its job without ever trashing its style by using and generating goto's for the compiled code. :-)

Best regards from V.

P.S.: Also, I happen to remember that in the realm of HP-41C synthetic programming, two of the most powerful synthetic instructions were the pair **STO b/RCL b**. You could do just about *everything* by cleverly using them, from creating any synthetic instruction or text as a program line, to byte-count a program, to create assignments, to defeat PRIVATE, you name it !

And guess what ? STO b **IS** a GOTO !

Good programming languages (S-OT)

Posted by ***John L. Shelton*** *on* ***20 May 2005, 2:21 p.m.****, in response to Re: Bad Languages (OT), posted by Valentin Albillo on 20 May 2005, 7:42 a.m.*

S-OT = "Still off-topic"

It's a mistake to think that because a compiler generates code of quality "X", that humans should write using the same quality "X". Otherwise, why use compilers at all? The point to a compiler is to allow us to write programs in something "expressive", better fitting our minds or the problem domain. Computers execute machine language designed to be efficient and general purpose.

There are many studies, some classic, demonstrating why the use of "goto" leads to more errors in human-written programs. In 25 years of progamming in Lisp, Smalltalk, and (yuk) Java, I haven't missed "goto".

What's a real shame is that "modern" languages like Java have been only small improvements over earlier languages, in terms of productivity. Java, according to several studies, is about 8 times as productive as writing in macro-assembly language. We ask our programmers today to do about 20 times as much functionality as we did a generation ago, and wonder why projects still take years. We need languages that are 100 times as productive (or customers with simpler project requirements.)

You are welcome to write machine-efficient code, for fun or profit. The skills required should not be forgotten - indeed, we need these skills inside new compilers. It's also fun, for nostalgia reasons, to program in primitive environments. I enjoy memories from the 1960s and 1970s, dealing with very small memory, very slow CPUs, and having to fine-tune things on paper before having a few seconds of time on the shared computer.

But for most modern programming today, I want very powerful languages that minimize human error, and good compilers to give me efficiency. But I value correctness and productivity much more than efficiency; I can buy more computer horsepower much more easily than buying developer time.

Some day, our computers will interview our customers, guided by good consultants, and the problem will be modeled correctly in the computer, with automatically generated programs appearing daily. As the customer plays with the incremental solutions, commenting on their merits and drawbacks, the consultant and computer will collaborate to generate the next versions.

Re: *Good* programming languages (S-OT)

Posted by ***Thomas Okken*** *on* ***24 May 2005, 7:48 a.m.****, in response to *Good* programming languages (S-OT), posted by John L. Shelton on 20 May 2005, 2:21 p.m.*

There are many studies, some classic, demonstrating why the use of "goto" leads to more errors in human-written programs. In 25 years of progamming in Lisp, Smalltalk, and (yuk) Java, I haven't missed "goto".

I suspect that those studies find what they want to find... Personally, I can get things done in Java, but I enjoy being able to use goto in C/C++, because it allows me to write code like this.

I'm sure some would consider this example to be a good case *against* the use of goto, but I find this kind of code easier to write *and* easier to read -- if you have to do this type of function without goto, you end up having to break the function into smaller ones, and/or introduce state variables and use lots of if statements, all of which makes the flow of control harder to follow.

Just my $0.02!

- Thomas

Re: *Good* programming languages (S-OT)

*Posted by **Valentin Albillo** on **24 May 2005, 9:13 a.m.**, in response to Re: *Good* programming languages (S-OT), posted by Thomas Okken on 24 May 2005, 7:48 a.m.*

Hi, Thomas:

Thomas posted:

"I suspect that those studies find what they want to find..."

Let's see. What follows is a small sample of random Intel assembler source code obtained from a random search using Google:

```
start:  mov ax, @data
        mov ds, ax
        mov ah, 9h            ;print help message
        mov dx, offset help
        int 21
hlp1:   mov ah, 06h           ; read character from keyboard
        mov dl, 0ffh
        int 21h
        jz lp1                ; repeat if character not ready
        cmp al, 00h           ; if function key then exit
        je exit
        cmp al, 32            ; else if control code
        jae disp1
        call ctrl_code        ; then process control code
        jmp lp1
disp1:  push bx
        xor bx, bx            ; page zero on video memory
        mov bl, [attrib]      ; get character attribute
        mov cx, 1             ; one character to write
        mov ah, 9             ; write char + attribute
        int 10h               ; use BIOS call
        call bumpcur          ; next cursor position
        jmp lp1               ; repeat
exit:   mov ax, 4c00h
        int 21h
END     start
```

I count 25 instructions (lines), and unless I'm mistaken, there are no less than 11 branching instructions that divert execution flow elsewhere. That's 44% for this particular, absolutely random sample. Even if you only want to count those branching instructions that do not return, there are still 5 of them (20%) in this very small piece of code.

"I suspect that you find unworthy those studies that do not find what you want to find..." :-)

Best regards from V.

Re: *Good* programming languages (S-OT)

*Posted by **.** on **24 May 2005, 6:45 p.m.**, in response to Re: *Good* programming languages (S-OT), posted by Valentin Albillo on 24 May 2005, 9:13 a.m.*

Yes, but 99% of people don't write in assembler. Those comments apply to structured programming languages.

.

Re: The Basket Case

*Posted by **Raymond Del Tondo** on **20 May 2005, 6:11 p.m.**, in response to Re: The Basket Case, posted by Valentin Albillo on 20 May 2005, 4:43 a.m.*

Hi,

> *Everything* is a mess in RPL
>
I totally disgree here!
But I think you meant your statement as a joke:-)

> Where you had the simplicity and extreme readability of BASIC
>
This is considered to be the 2nd part of the kidding statement...

Even simple BASIC programs can be relatively unreadable.
I recently saw such an example in datafile, right in the middle;-)

Someone tried to squish as many as possible statements into one line,
thus reducing the program size by a few bytes,
but abandonning readability totally IMHO.

Back then when I programmed the 71B in BASIC,
I even put local labels in the middle of lines,
just to save the byte or two required for the line number.

I loved these byte saving techniques a lot,
and still love to optimize programs in size and speed,
but I also learned to love structured source text.

You could write a complex RPL program in one single line, too,
but it could get as unreadable as said BASIC program.

With RPL, you have the choice:
Writing BASIC-like spaghetti code,
which will be unreadable to yourself after a while,
unless you provide a good documentation,
or writing structured code,
where you can at least recognize the bigger
functional blocks even after months.

BTW: In RPL, you don't need a an explicit GOTO statement,
because you have other nice ways to transform
program execution to somewhere else.

And RCL b/STO b on the HP-41 are nice and fast GTO alternatives,
but they're far away from being even considered as clean programming style.
They can even be dangerous,
and thus I don't recommend the use of RCL b/STO b,
especially if you give programs to someone else,
who doesn't know how to handle RCL b/STO b and alike.

Regards Raymond

Re: The Basket Case

*Posted by **Valentin Albillo** on **23 May 2005, 5:31 a.m.**, in response to Re: The Basket Case, posted by Raymond Del Tondo on 20 May 2005, 6:11 p.m.*

Hi, Raymond:

Raymond posted:

"I think you meant your statement as a joke:-)"

Not exactly. I don't recall adding a smiley after my statement.

"Even simple BASIC programs can be relatively unreadable. I recently saw such an example in datafile, right in the middle;-). Someone tried to squish as many as possible statements into one line, thus reducing the program size by a few bytes, but abandonning readability totally IMHO."

I *do* notice your 'sarcastic' emoticon, which I take it to mean you're referring to some program of mine published in Datafile. If that's so, you're absolutely wrong. I do **not** *"squish as many as possible statements into one line"* to reduce *program* size but to reduce *article* size.

Datafile is a small-format publication, just 32-48 A5 pages, published every other month, so saving space is *critical*, in my opinion, to try and fit as many interesting articles as possible from as many contributors as possible. Were I to publish my program listings in 'pretty printing' format, with every statement on its own line, indentations and such, it would take up to 5 times more valuable space in Datafile for the same net article content. I would never hog Datafile's space that way, publishing 'pretty looking' listings at the expense of not having space for other people's routines.

As a simple example, my solution to S&SMC#9 could be published there in 4 lines, thus:

```
    100  SUB SR(N,L) @ FOR I=MOD(L,2)+4*(L=5) TO 9 STEP 2+3*(L=5)
    110  IF FLAG(I) THEN 130 ELSE M=N+I @ IF MOD(M,L) THEN 130
    120  SFLAG I @ IF L<9 THEN CALL SR(10*M,L+1) @ CFLAG I ELSE DISP M*10 @
CFLAG I
    130  NEXT I
```

or else, 'pretty formatted' this way:

```
    100  SUB SR(N,L)
    105      FOR I=MOD(L,2)+4*(L=5) TO 9 STEP 2+3*(L=5)
    110          IF FLAG(I) THEN
    112              GOTO 130
    114          ELSE
    116              M=N+I
    118              IF MOD(M,L) THEN GOTO 130
    119          END IF
    120          SFLAG I
    122          IF L<9 THEN
    123              CALL SR(10*M,L+1)
    124              CFLAG I
    125          ELSE
    126              DISP M*10
    127              CFLAG I
    128          END IF
    130      NEXT I
    135  END SUB
```

taking 5 times as much space in the page. When you've got only 32-48 small (A5) pages for *everyone, every two months*, hogging 400% more space is unacceptable, specially when your intended audience (Datafile readers) are *very* skilled people , used to program HP calculators, and who won't have any difficulty whatsoever understanding your listing or even formatting it to their own taste if desired.

You're a Datafile subscriptor, I take, so tell me if you'd rather have Datafile space used up to include some interesting RPL routines or else to have instead Mr. Albillo's program for the HP-71B 'pretty-printed', with no statements "squished into one line".

"Back then when I programmed the 71B in BASIC, I even put local labels in the middle of lines, just to save the byte or two required for the line number."

Correct me if I'm wrong, but I think that any "local label" (say "PEPE") and its correspondig GOTO (GOTO "PEPE") will waste many more bytes than the two required for the line number, right ?

"You could write a complex RPL program in one single line, too, but it could get as unreadable as said BASIC program."

I don't think you're being serious here, or else you're confusing semantic unreadability with sintactic unreadability. Any BASIC code is highly human-readable for English readers because it uses mostly English keywords with an English-like syntax. You can put evey word in a single line, or a lot of words on a line, or even supress spaces between words, yet it will still be mostly readable and understandable, say for instance:

```
   FOR Quantity = 10 TO 20
       LET Cost = Price * Quantity
       PRINT Price, Quantity, Cost
   NEXT Quantity
```

or even not formatted at all

```
   FOR Quantity = 10 TO 20 : LET Cost = Price * Quantity :  PRINT Price,
Quantity, Cost : NEXT Quantity
```

pose no understanding problems for any English reader, be he/she a BASIC programmer or not. On the other hand, RPL code such as:

```
<< Cucu a 1
rdPOS SUB SWAP 1 \->LIST + a 'rdPOS'
INCR rdSIZ SUB +
'rdNAME' RCL STO
rdINIT \>>
```

is inherently very difficult to fathom, however formatted, for most everyone, including RPL programmers themselves. So I don't think your next paragraph is to be taken seriously at all, but that's your opinion and this is *my* opinion. Let's stop here and avoid flame wars and other childish behaviour, such quasi-religious topics can't be discussed profitably and I know from experience that trying to change other people's deeply-rooted beliefs, however irrational they might be, is doomed to bitter failure.

Best regards from V.

Edited: 23 May 2005, 5:41 a.m.

S&SMC#9: My Solutions and Comments

*Posted by **Valentin Albillo** on **20 May 2005, 6:36 a.m.**, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Hi, all:

Thanks to all people interested in my S&SMC#9, there weren't a lot of entries but the ones posted were extremely interesting and accurate, my sincerest congratulations to the contributors. These are my original solutions:

- **Sub-Challenge (1):**

  This kind of problem is ideally suited for a recursive solution and thankfully HP-71B's powerful BASIC dialect allows for it. My solution is a 4-line recursive subprogram just 136 bytes long. All its 'smarts' are encapsulated in the careful setup of the FOR statement and the recursivity itself:

```
    100  SUB SR(N,L) @ FOR I=MOD(L,2)+4*(L=5) TO 9 STEP 2+3*(L=5)
    110  IF FLAG(I) THEN 130 ELSE M=N+I @ IF MOD(M,L) THEN 130
    120  SFLAG I @ IF L<9 THEN CALL SR(10*M,L+1) @ CFLAG I ELSE DISP
M*10 @ CFLAG I
    130  NEXT I
```

  Executing it from the keyboard, produces the only solution:

```
    >CFLAG ALL @ CALL SR(0,1)

       3816547290
```

  in 28 seconds (0.32 sec in Emu71 @ 2.4 Ghz)

- **Sub-Challenge (2):**

  Likewise, this is best solved by a slight variation of the above subprogram, the following 4-line recursive subprogram (149 bytes, notice the FOR loop is set up *differently* than in the previous subprogram):

```
    100  SUB SR(N,L,H,C) @ FOR I=L=1 TO 9 STEP 2-MOD(L,2)+4*(L=5)
    110  IF FLAG(I) THEN 130 ELSE M=N+I @ IF MOD(M,L) THEN  130
    120  SFLAG I @ IF L<H THEN CALL SR(10*M,L+1,H,C) @ CFLAG I ELSE
C=C+1 @ CFLAG I
    130  NEXT I
```

  Calling it from the keyboard produces a list of the number of solutions for each N (706 in all):

```
    >FOR I=1 TO 10 @ CFLAG ALL @ C=0 @ CALL SR(0,1,I,C) @ DISP I;C @
NEXT I

  1  9
  2  41
  3  115
  4  168
  5  220
  6  88
  7  51
  8  9
  9  4
 10  1
```

  in just 25 minutes (7 seconds in Emu71 @ 2.4 Ghz)

- **Sub-Challenge (3):**

  Again, some variation does the job, this time an even simpler and *shorter* (3-line, 115 byte) recursive subprogram:

```
    100  SUB SR(N,L,H,C) @ FOR I=L=1 TO 9 STEP 2-MOD(L,2)+4*(L=5) @
M=N+I
    110  IF MOD(M,L) THEN 120 ELSE IF L<H THEN CALL SR(10*M,L+1,H,C)
ELSE C=C+1
    120  NEXT I
```

  Executing it from the keyboard produces the list of the number of solutions (11,453 in all):

```
    >FOR I=1 TO 10 @ CFLAG ALL @ C=0 @ CALL SR(0,1,I,C) @ DISP I;C @
NEXT I

  1  9
  2  45
  3  150
  4  375
  5  750
  6  1200
  7  1713
  8  2227
  9  2492
 10  2492
```

  in some 3 hours (only 10 min in Emu71 @ 2.4 Ghz).

  Comments:

  However it might seem, these subprograms are anything but optimized, according to my philosophy on how these challenges should be met. There are many things that can be done to speed them up, so way to go.

  For case (3), though the number of solutions seems to grow forever that's not the case. Actually, 2492 is the maximum number of solutions for any N (in this case, for N=9 and N=10).

  After that (N=11, 12, etc), the number of solutions decreases steadily till N=25, which again has a unique, 25-digit solution. Since none of the digits 0-9 results in a number divisible by 26 when added, there are no solutions for N=26 and beyond.

Thanks for your interest, see you all in a future S&SMC#10.

Best regards from V.

Edited to remove a couple typos

Edited: 20 May 2005, 6:59 a.m.

Re: Short & Sweet Math Challenge #9: Divisibility !

*Posted by **Chris Dean** on **20 May 2005, 11:45 a.m.**, in response to Short & Sweet Math Challenge #9: Divisibility !, posted by Valentin Albillo on 16 May 2005, 9:31 a.m.*

Before you close I have a 49g+ solution which uses only the simplest rules for the odd and even 5 and 0 placings of the digits. The program was very laborious to write but gets the solution as required. It depends on the view point of the solver whether they are interested in the solution or elegance of the software solution. The choice is for the programmer only. The code would have been optimised but firstly I was interested in getting the solution. here it is

```
<< 1 9 FOR A << 2 8 FOR B << 1 9 FOR C IF 100 A * 10 B * + C + 3 MOD 0 == THEN << 2
8 FOR D IF 1000 A * 100 B * + 10 C * + D + 4 MOD 0 == THEN << 2 8 FOR E IF 100000 A *
10000 B * + 1000 C * + 100 D * + 50 + E + 6 MOD 0 == THEN << 1 9 FOR F IF 1000000 A 8
100000 B * + 10000 C * + 1000 D * + 500 + 10 E * + F + 7 MOD 0 == THEN << 2 8 FOR G IF
10000000 A * 10000000 B * + 1000000 C * + 100000 D * + 5000 + 100 E * + 10 F * + G 8
MOD 0 == THEN << 1 9 FOR H IF 100000000 A * 100000000 B * + 10000000 C * + 1000000
D * + 50000 + 1000 E * + 100 F * + 10 G * + H + 9 MOD 0 == THEN IF ‘A<>5 AND C<>5
AND F<>5 AND H<>5 AND A<>C AND A<>F AND A<>H AND C<>F AND C<>H AND
F<>H AND B<>D AND B<>E AND B<>G AND D<>E AND D<>G AND E<>G’ THEN
1000000000 A * 100000000 B * + 10000000 C * + 1000000 D * + 500000 + 10000 E * + 1000
F * + 100 G * + 10 H * END END 2 STEP >> ->NUM END 2 STEP >> ->NUM END 2 STEP
>> ->NUM END 2 STEP >> ->NUM END 2 STEP >> ->NUM END 2 STEP >> ->NUM END
2 STEP >> ->NUM END 2 STEP >> ->NUM
```

The <> is for 'not equals'. As a programmer I can rip this software solution to shreds for inefficiency but as stated earlier I was more interested in the result and it does that job yielding 3816547290.

Re: Short & Sweet Math Challenge #9: Divisibility ! (with formatted RPL listing)

*Posted by **Jeff O.** on **20 May 2005, 11:58 a.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility !, posted by Chris Dean on 20 May 2005, 11:45 a.m.*

begging your permission, or at least your pardon, here's the listing with the formatting that I believe you intended:

Before you close I have a 49g+ solution which uses only the simplest rules for the odd and even 5 and 0 placings of the digits. The program was very laborious to write but gets the solution as required. It depends on the view point of the solver whether they are interested in the solution or elegance of the software solution. The choice is for the programmer only. The code would have been optimised but firstly I was interested in getting the solution. here it is

```
<< 1 9 FOR A
   << 2 8 FOR B
      << 1 9 FOR C
      IF 100 A * 10 B * + C + 3 MOD 0 == THEN
         << 2 8 FOR D
         IF 1000 A * 100 B * + 10 C * + D + 4 MOD 0 == THEN
             << 2 8 FOR E
             IF 100000 A * 10000 B * + 1000 C * +
            100 D * + 50 + E + 6 MOD 0 == THEN
                 << 1 9 FOR F
                  IF 1000000 A 8 100000 B * + 10000 C * + 1000 D * +
                  500 + 10 E * + F + 7 MOD 0 == THEN
                  << 2 8 FOR G
                  IF 10000000 A * 10000000 B * + 1000000 C * +
                 100000 D * + 5000 + 100 E * + 10 F * + G 8 MOD 0 == THEN
                     << 1 9 FOR H
                     IF 100000000 A * 100000000 B * + 10000000 C * +
                    1000000 D * + 50000 + 1000 E * + 100 F * +
                    10 G * + H + 9 MOD 0 == THEN
                    IF ‘A<>5 AND C<>5 AND F<>5 AND H<>5 AND
                         A<>C AND A<>F AND A<>H AND C<>F AND
                         C<>H AND F<>H AND B<>D AND B<>E AND
                         B<>G AND D<>E AND D<>G AND E<>G’ THEN
                        1000000000 A * 100000000 B * + 10000000 C * +
                        1000000 D * + 500000 + 10000 E * + 1000 F * +
                        100 G * + 10 H *
                     END
                     END 2 STEP >> ->NUM
                  END 2 STEP >> ->NUM
               END 2 STEP >> ->NUM
            END 2 STEP >> ->NUM
         END 2 STEP >> ->NUM
      END 2 STEP >> ->NUM
   END 2 STEP >> ->NUM
END 2 STEP >> ->NUM
```

The <> is for 'not equals'. As a programmer I can rip this software solution to shreds for inefficiency but as stated earlier I was more interested in the result and it does that job yielding 3816547290.

Edited: 20 May 2005, 11:59 a.m.

Re: Short & Sweet Math Challenge #9: Divisibility ! (with formatted RPL listing)

*Posted by **Chris Dean** on **20 May 2005, 2:34 p.m.**, in response to Re: Short & Sweet Math Challenge #9: Divisibility ! (with formatted RPL listing), posted by Jeff O. on 20 May 2005, 11:58 a.m.*

Jeff

Thanks for the formatting. That is what I did in the word document I produced!!

Thanks again

Chris Dean

Re: S&SMC#9-Question to Valentin

Posted by ***Bill (Smithville, NJ)*** *on* ***23 May 2005, 7:32 a.m.***

Hi Valentin,

I feel I must gently chastise you a little bit - hopefully in good humor. :)

When I initially read your description of the challenge and the rules you laid down I thought I'd get to see another one of your well thought out and documented RPN calculator solutions. Something similar to your Datafile articles. Your rules stated:

"Giving *just* the solutions (but no program) or providing programs for machines other than (preferably HP) *calculators* is to be considered as blatant disrespect to the stated rules and/or a clear statement of utter inability to comply within these terms."

The word calculator was underlined to stress that you were looking for calculator solutions. You then proceeded to state:

"Two or three days from now I'll post my solution (plus comments), which is essentially a rather simple, unoptimized 4-line program (plus variations) for a bare-bones HP-71B that takes less than 2 minutes to find the only solution..."

I realize that you view the HP-71B as a "calculator" but I would view it more as a "computer" since it uses Basic. I could easily use Basic on my PC and then translate it to the HP-71B or one of the Sharp versions of Basic. But wouldn't you say that would defeat the purpose of a HP "calculator" challenge.

Please take my comments as a gentle nudge - I always enjoy seeing your HP-71B solutions, BUT I much more enjoy the RPN solutions. Fortunately there was one RPN solution but it would have been great to see yours also.

I'm afraid I didn't have much luck with this particular challenge, but enjoyed it anyway. I'm looking forward to trying my luck with your next challenge.

Bill

12345

Re: S&SMC#9-Question to Valentin

*Posted by **Valentin Albillo** on **23 May 2005, 9:55 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Bill (Smithville, NJ) on 23 May 2005, 7:32 a.m.*

Hi, Bill:

Bill posted:

"I feel I must gently chastise you a little bit - hopefully in good humor. :)"

Be my guest, let's see ...

"When I initially read your description of the challenge and the rules you laid down I thought I'd get to see another one of your well thought out and documented RPN calculator solutions [...] The word calculator was underlined to stress that you were looking for calculator solutions."

That's correct.

"I realize that you view the HP-71B as a "calculator" but I would view it more as a "computer" since it uses Basic."

Yes, it's the same old problem discussed many times before of the exact definition of "calculator" vs. "computer". The extreme cases are obvious, but there's a certain blurred line where you can't assign a clear classification. For some, the HP-71B is a computer, while the HP-41C is a calculator. Yet both are the central unit of complete systems, do have CPU, display, RAM, ROM, peripherals, system extensions, you name it. Why should one be considered a "calculator" and the other a "computer" ? No reason at all. Even HP themselves called the 41C a "computer" in a lot of marketing adds and materials.

Of course, I could clarify by giving a large list of allowed machines, but that's nonsensical. We all know what kind of machines we're talking about. An HP-25 ? Yes. HP-41C ? Yes. HP-71B ? Yes. HP-48/49 (much more powerful than the 71B) ? Yes. SHARP PC-1211 ? Yes. TI-59 ? Yes.

A Pentium-based PC compatible ? No. Some mainframe ? No. An ancient Apple II running BASIC ? No. A Sinclair ZX Spectrum ? No. Some PDA ? No. I hope you get the point.

"I could easily use Basic on my PC and then translate it to the HP-71B or one of the Sharp versions of Basic."

By all means do. And don't forget to post the resulting program.

"But wouldn't you say that would defeat the purpose of a HP "calculator" challenge."

No. As long as the resulting program runs in one of 'approved' models, be it an HP-41, HP-71, or HP-48, say, I don't mind how it came into existence, be it as a translation from a Cray-1 program, or because you channeled some alien in Omicron Persei 8.

"Please take my comments as a gentle nudge - I always enjoy seeing your HP-71B solutions, BUT I much more enjoy the RPN solutions."

Then submit one yourself, so that everyone can see how it's done.

"Fortunately there was one RPN solution but it would have been great to see yours also."

Thanks a lot for your interest and kind words, but giving MY solutions for the HP-71B instead of for some RPN (or RPL) model has a number of *important* advantages, among them:

1. Everyone can try out a solution for the HP-71B, as there is a perfect emulator, Emu-71, which everyone can download and use for free, to test my solutions, or to develop their owns. Should I give a solution for other machines, there may or may not exist a free emulator (including free ROMs) for that machine, or people may not have that physical model.

2. HP-71B's BASIC is powerful enough that a solution can be given in a (very) few lines, and being BASIC, it's easy to understand the code, and thus easy to translate to other calculator languages. An RPN solution would typically required a hundred steps or more, and it's much more cryptic and difficult to understand, let alone translate it to run in other machines, even if RPN.

3. It's very easy for me to copy-paste the program listing and results from Emu71. Doing it from a physical machine or other emulators would be a lot more work and error prone, and designing these challenges and creating the solutions already requires more free time that I can reasonable allocate to this, so the less unnecessary work involved, the better.

... and there are more reasons, but I think these three make my point clear.

"I'm afraid I didn't have much luck with this particular challenge, but enjoyed it anyway. I'm looking forward to trying my luck with your next challenge."

... and I'm looking forward for your next contribution, be it an RPN program or some IBM PC Basic program you translated to HP-71B BASIC (or RPL, for that matter). And of course, submitting an *untranslated* program would be utterly unacceptable. :-)

Thanks a lot for your interest and comments, and

Best regards from V.

Re: S&SMC#9-Question to Valentin

Posted by ***.*** *on* ***23 May 2005, 10:26 a.m.****, in response to Re: S&SMC#9-Question to Valentin, posted by Valentin Albillo on 23 May 2005, 9:55 a.m.*

>Of course, I could clarify by giving a large list of allowed machines, but that's nonsensical. We all know what kind of machines we're talking about. An HP-25 ? Yes. HP-41C ? Yes. HP-71B ? Yes. HP-48/49 (much more powerful than the 71B) ? Yes. SHARP PC-1211 ? Yes. TI-59 ? Yes. <> A Pentium-based PC compatible ? No. <> Some PDA ? No. I hope you get the point. ===

But what is the difference? You can write a program in a PC systems language, and very easily recompile it for a Texas or a 49 with minimal changes. Aside from the smaller RAM and slower CPU (not really an issue in most of the challenges) the program will execute identically on both.

Suppose I developed and debugged my program in a PC, and then simply recompiled it for a HP49 (maybe ten mins work). Does that qualify as a valid solution? Is so, where is the challenge? It is far less work for me than for someone using an ancient HP.

Modern calculators are just PDAs anyway, with less RAM and a different UI.

.

Re: S&SMC#9-Question to Valentin

*Posted by **Valentin Albillo** on **23 May 2005, 11:00 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by . on 23 May 2005, 10:26 a.m.*

Hi, . :

. posted:

"You can write a program in a PC systems language, and very easily recompile it for a Texas or a 49 with minimal changes. Aside from the smaller RAM and slower CPU (not really an issue in most of the challenges) the program will execute identically on both."

I think my answer to Bill was clear enough: as long as you post a solution that will *run* in some HP calc model (or even some suitably ancient SHARP or other brands), I don't care how you managed to create it. You want to do it as it is intended, by writing a program directly for a physical HP calc using a physical HP calc ? Good. You want to do it in a roundabout way by first writing it in, say, Java, on a laptop, then translating that Java code to RPN ? Equally good, you'll probably enjoy doing it and learn something in the process.

As long as you then post a solution in RPN, RPL, 71B BASIC, ancient SHARP handhelds BASIC, TI's AOS, or any other language for any other model according to the examples I gave, I don't mind. It's the end result that matters to me, not how you got there.

Myself, I originally write the solutions in HP-71B BASIC using Emu71, for the reasons given in my answer to Bill. If the challenge is for the HP-15C, I might use either a real 15C or an emulator, but I´ll post something that *runs* on a *physical* HP-15C, even if I designed and tested the original algorithm in Java, say. Mind you, translating that Java code to 15C's RPN would be a major programming feat in itself, don't you think ?

"It is far less work for me than for someone using an ancient HP."

Good for you. Go ahead full steam and please, post your results.

"Modern calculators are just PDAs anyway, with less RAM and a different UI."

That's an opinion, not necessarily a fact. I'm not very interested in modern calculators. This is the "*Museum* of HP"'s Forum, so ancient models should be the primary focus, you can always use comp.sys.hp48 for modern models. My challenges are thus primarily intended for ancient models, though I'm pretty happy to get solutions for modern machines such as the 48/49, if only for comparison purposes. I also try to delve in curious or otherwise interesting mathematical facts as well.

Anyway, as long as people enjoy the challenges and, ideally, learn something or find something useful, my goal is met. I don't really care for strict rules, limits or exclusions, but I think that giving a Java solution with timings acquired on a Pentium IV at 3.2 Ghz for a 4-line challenge intended for ancient calculators is simply missing the mark by a parsec or so.

Thanks for you interest and comments, and

Best regards from V.

Edited: 23 May 2005, 11:02 a.m.

Re: S&SMC#9-Question to Valentin

*Posted by **.** on **23 May 2005, 11:16 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Valentin Albillo on 23 May 2005, 11:00 a.m.*

>You want to do it in a roundabout way by first writing it in, say, Java, on a laptop, then translating that Java code to RPN ? Equally good, you'll probably enjoy doing it and learn something in the process.

My point is you don't need to translate. These days the same languages work perfectly fine on both PCs and calculators. It is almost trivial to port code from one to the other.

Whats the command to print 'Hello World' on a TI89?

printf("Hello World");

What about on a casio?

printf("Hello World");

or a HP49?

printf("Hello World");

Sure, some things are different (such as hardware specific to each model). The underlying language can remain constant.

>Modern calculators are just PDAs anyway, with less RAM and a different UI."

That's an opinion, not necessarily a fact ==

The HP49 shares the same CPU as HP's PDAs. Casio AFXs have an '86 CPU that run DOS. The Casio Classpad has a far faster CPU than my first computer. The Saturn is dead, long live the Saturn.

>but I think that giving a Java solution with timings acquired on a Pentium IV at 3.2 Ghz for a 4-line challenge intended for ancient calculators is simply missing the mark by a parsec or so.

Of course it is. But you can write, compile and debug a program on a desktop PC, and use high quality tools to do it. Once you are satisified that it works, it is a matter of minutes to port it to a calculator and run it. It's clearly easier than trying to write and RPN program on a model with limited memory.

Best wishes,

.

Re: S&SMC#9-Question to Valentin

*Posted by **Arnaud Amiel** on **23 May 2005, 12:25 p.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by . on 23 May 2005, 11:16 a.m.*

For the newest calculators, I believe the challenge should be that the program should be input as a source on the calculator and compiled with the tools included with the calculator. I was thinking of doing MC9 in ARM assembly but didn't due to time and most importantly because you can't do it directly on the calculator with the built in tools.

For me, if you can enter the program from the source on the calculator you are OK. Now we just need a calculator with a built in C compiler. (I believe CASIO produced a few)

Arnaud

Re: S&SMC#9-Question to Valentin

*Posted by **.** on **23 May 2005, 7:16 p.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Arnaud Amiel on 23 May 2005, 12:25 p.m.*

I think that is the fairest solution.

.

Re: S&SMC#9-Question to Valentin

*Posted by **Bill (Smithville, NJ)** on **23 May 2005, 1:13 p.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Valentin Albillo on 23 May 2005, 9:55 a.m.*

Hi Valentin,

Thank you for your detailed reply.

I guess I differentiate between what I consider a calculator and the small pocket computers which use Basic as their primary language. I love using both. The main difference between programming a problem in Basic on the Pocket PC as opposed to programming it on a full size PC is one of keyboard and screen size. Both have similar instruction sets. I guess I'm missing what the challenge is in using Basic to solve the problem - other than the fun of typing it on a small keyboard :)

You state that by using the HP-71B, more people could run your solution and that the Basic is easier to understand. I appreciate both arguments. And I can appreciate your time constraints - I also have trouble finding free time to play with my HP's.

But there are several HP-41 emulators, a HP-42S emulator, and several suimulators - plenty of ways for people to check out a RPN program.

Since I use my HP's on a daily basic, I'm always looking for programs and algorithms to analyze so that I can improve my RPN programming capability. From your Datafile articles, I know how good your RPN programming is and was hoping I could analyse your solutions to the challengers.

"Then submit one yourself, so that everyone can see how it's done."

As I said, I didn't have much luck with solving it. If I had, I would have definetly posted it.

Still working on the current challenge and looking forward to the next one.

Bill

12345

Re: S&SMC#9-Question to Valentin

*Posted by **Valentin Albillo** on **24 May 2005, 4:46 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Bill (Smithville, NJ) on 23 May 2005, 1:13 p.m.*

Hi again, Bill:

Bill posted:

"The main difference between programming a problem in Basic on the Pocket PC as opposed to programming it on a full size PC is one of keyboard and screen size. Both have similar instruction sets."

I see you've done very little programming in HP-71B's BASIC. If that's the case, believe me, your "similar instructions sets" couldn't be farther from the truth. When compared to typical BASICs of its time (say, Microsoft BASIC, IBM's BASICA, or GW-BASIC), it's light-years ahead from them, specially when you take into account the powerful extensions to its already awesome instruction set afforded by such plugs-in as the Math ROM, HP-IL ROM, etc.

As such, programming most nearly everything in HP-71B's BASIC is a unique, enthralling experience, where sheer power and, paradoxically, utter simplicity become so intimately intertwined as to make it all a most rewarding activity. Gone are the struggles to try and fit everything within the constraints of RPN. Gone are the headaches associated with trying to use RPL, where there are simply so many available functions, types, objects, structures, everyone of them with their different parameters, quirks, response to flags, etc, that you simply lose focus.

As you may have seen, most of my challenges, even the ones you find difficult to solve, can be dealt with in 3,4, or 5 lines of HP-71B BASIC. Recently, one of my published articles in Datafile showed how to solve a complicated engineering-math problem (finding all eigenvalues, real or complex, of an arbitrary NxN square matrix) in exactly 5 lines of code.

Just try it in MS BASIC, BASICA, GW-BASIC, or even Visual BASIC .NET, for that matter. You'll be amazed at the enormous difficulty of the task, and the number of lines of code it takes.

By the way, this doesn't mean I'll limit my challenges and/or solutions just to the 71B, of course. I'll also publish RPN-specific challenges and will give RPN solutions when time permits.

Best regards from V.

Re: S&SMC#9-Question to Valentin

*Posted by **Marcus von Cube** on **24 May 2005, 7:03 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Valentin Albillo on 24 May 2005, 4:46 a.m.*

> Quote:
>
> ---
>
> As you may have seen, most of my challenges, even the ones you find difficult to solve, can be dealt with in 3,4, or 5 lines of HP-71B BASIC.
>
> ---

Valentin, what I particularly don't like in your solutions is the fact that you tend to fill the lines with as many instructions as fit. I always try to write code in a way that is not only legible to computers but to me as a human as well.

So 3 to 4 lines will easily become 20 to 30 instructions. Still much less than any keystroke program and still much easier to understand than any RPL program (I gave up on my 48) regardless of formatting.

I love the challenges but I don't have the time to do them all (I posted what I believe could be a working algorithm, at least).

Keep on challenging!

S&SMC#9: My ** RPN ** solution & comments

*Posted by **Valentin Albillo** on **25 May 2005, 5:05 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Bill (Smithville, NJ) on 23 May 2005, 1:13 p.m.*

Hi again, Bill:

Bill posted:

"I'm always looking for programs and algorithms to analyze so that I can improve my RPN programming capability. From your Datafile articles, I know how good your RPN programming is and was hoping I could analyse your solutions to the challengers."

Lest you think I post my solutions in 71B's BASIC out of laziness or RPN shyness, here's a direct translation of my original 71B solution to case (1), namely:

```
  100  SUB SR(N,L) @ FOR I=MOD(L,2)+4*(L=5) TO 9 STEP 2+3*(L=5)
  110  IF FLAG(I) THEN 130 ELSE M=N+I @ IF MOD(M,L) THEN 130
  120  SFLAG I @ IF L<9 THEN CALL SR(10*M,L+1) @ CFLAG I ELSE DISP M*10 @
CFLAG I
  130  NEXT I
```

to genuine, HP-42S RPN code:

```
  01*LBL "SR"   24  1           47 RCL 00         70 RCLx IND 03
  02 CF 00      25  +           48 2              71 STOP
  03 CF 01      26 STO 03       49 MOD            72 GTO 04
  04 CF 02      27 X<>Y         50 +              73*LBL 02
  05 CF 03      28 STO IND 01   51 STO IND 02     74 10
  06 CF 04      29 RCL 00       52*LBL 01         75 RCLx IND 03
  07 CF 05      30  5           53 RCL IND 02     76  1
  08 CF 06      31 CF 15        54 FS? IND ST X        77 RCL+ 00
  09 CF 07      32 X=Y?         55 GTO 03         78 XEQ 00
  10 CF 08      33 SF 15        56 IP             79  1
  11 CF 09      34  3           57 RCL+ IND 01    80 STO- 00
  12  0         35 FC? 15       58 STO IND 03     81  3
  13  1         36  0           59 RCL 00         82 STO- 01
  14*LBL 00     37  2           60 MOD            83 STO- 02
  15 STO 00     38  +           61 X#0?           84 STO- 03
  16  3         39 1E5          62 GTO 03         85*LBL 04
  17  x         40  /           63 RCL IND 02     86 RCL IND 02
  18  1         41 0.009        64 SF IND ST X    87 CF IND ST X
  19  +         42  +           65  9             88*LBL 03
  20 STO 01     43  4           66 RCL 00         89 ISG IND 02
  21  1         44 FC? 15       67 X<Y?           90 GTO 01
  22  +         45 CLX          68 GTO 02         91 RTN
  23 STO 02     46 +            69 10             92 END
```

To run it, simply set SIZE 31 or greater, and:

```
   XEQ "SR"   ->   3,816,547,290
```

you'll get the first (and only) answer in a few seconds. As stated, this program is a direct translation of the original, which is slightly longer (167 bytes vs. 136 bytes for the original), though that's irrelevant as it can easily be optimized to shave off 10-20 program steps or more. I refrained from doing it to preserve the parallelism to the original for didactic comparison purposes, and as it runs so fast anyway.

Seeing this RPN code, there are some points I want to make, to further expand what I told you in my answer to your posting of a few days ago:

- The 71B's BASIC code is just four lines, so very compact and extremely easy to grasp at a glance and understand what it is doing and how it does work. On the other hand, the

RPN code is much longer, at 92 steps, and only experienced RPN programmers can eventually understand what it is doing, and it probably would take some step-by-step analysis to clarify it all (assuming, of course, that you're *not* aware of the 71B's version, that is !).

- The 71B's BASIC code, being so short and clear, can be easily translated to any other calculator languages, such as RPN (this exact case), RPL, or whatever. Trying to do the same starting from the RPN version is much more troublesome and a lot more work.
- It took me 5 minutes to write the 71B version, including testing and timing it. Copying it for posting was a case of copy & paste and it took just seconds. On the other hand, it took me more than an hour to write, debug, and test the RPN version, even though I was merely translating from the existing BASIC version, and preparing it for posting here meant to painstakingly write down the steps one by one, manually, and double-checking for accuracy, thus requiring another late night hour.

So I absolutely rest my case: it's a much more efficient use of my limited resources to give my solutions in 71B code and it's much more convenient for the readers, who can then adapt them to their favorite language. Of course, out of sheer human laziness, one would prefer to directly get the solutions in RPN, RPL, or whatever, but be a sport and do some little work yourself, I'm already doing my part, don't you think ?

"Still working on the current challenge and looking forward to the next one."

Thanks for your interest and please do try and post your findings. Remember, it's much easier to come up with some pretty reasonable excuses than to come up with some reasonably pretty solutions (the line's mine). :-)

Best regards from V.

Edited: 25 May 2005, 7:50 a.m.

Re: S&SMC#9: My ** RPN ** solution & comments

Posted by ***Olivier De Smet*** *on* ***25 May 2005, 8:03 a.m.****, in response to S&SMC#9: My ** RPN ** solution & comments, posted by Valentin Albillo on 25 May 2005, 5:05 a.m.*

Just one question, is there enough return level in the HP41 to allow 9 (or 8 ?) deep XEQ 00 ?

BTW here follow an HP86B version of your program (just have to tackle with local variables and recursivity because the HP86 basic does not allow recursivity on user FN). This could be also usefull for Sharp basic ...

```
list
10 SFLAG "" @ DIM n(9),i(9)
20 t=TIME  @ l=1 @ n(l)=-1 @ GOSUB 100
30 DISP "fini ";TIME -t @ END
100 i=1+l MOD 2+4*(l=5)
110 IF FLAG (i) THEN 160
120 m=n(l)+i @ IF m MOD l THEN 160
130 SFLAG i @ IF l=9 THEN DISP m*10 @ GOTO 150
140 i(l)=i @ l=l+1 @ n(l)=m*10-1 @ GOSUB 100 @ l=l-1 @ i=i(l)
150 CFLAG i
160 i=i+2+3*(l=5) @ IF i<= 10 THEN 110
170 RETURN
 517748
•
```

Olivier

Edited: 25 May 2005, 8:11 a.m.

Re: S&SMC#9: My ** RPN ** solution & comments

*Posted by **Valentin Albillo** on **25 May 2005, 9:07 a.m.**, in response to Re: S&SMC#9: My ** RPN ** solution & comments, posted by Olivier De Smet on 25 May 2005, 8:03 a.m.*

Hi, Olivier:

Olivier posted:

"Just one question, is there enough return level in the HP41 to allow 9 (or 8 ?) deep XEQ 00 ?"

No. Were it not for that fact, my RPN version would work unchanged on the HP-41C, except for the fact that all RCL arithmetic would have to be 'expanded', i.e.: instead of RCLx IND 03 you would have RCL IND 03, *.

The HP-15C does have RCL arithmetic, but also can't go deep enough and there's the problem of flags 8 and 9 being 'system flags', i.e. controlling special behavior. Anyway, it is entirely possible to overcome these minor difficulties and come up with a 41C or 15C version.

Thanks for your HP86B version, Olivier.

Best regards from V.

Re: S&SMC#9: My ** RPN ** solution & comments

*Posted by **Olivier De Smet** on **25 May 2005, 10:24 a.m.**, in response to Re: S&SMC#9: My ** RPN ** solution & comments, posted by Valentin Albillo on 25 May 2005, 9:07 a.m.*

Yes your program need exactly 8 levels and the HP42S got exactly 8 levels too :). The HP41 only remember 6 levels of sub-routine :(

Olivier

Re: S&SMC#9: My ** RPN ** solution & comments

Posted by ***Bill (Smithville, NJ)*** *on* ***25 May 2005, 10:00 a.m.****, in response to S&SMC#9: My ** RPN ** solution & comments, posted by Valentin Albillo on 25 May 2005, 5:05 a.m.*

Hi Valentin,

"Lest you think I post my solutions in 71B's BASIC out of laziness or RPN shyness, here's a direct translation of my original 71B solution to case (1), namely:"

With the amount of time you spend posting, replying to posts, creating challenges, articles for Datafile, and I assume you also have a job that requires your time, I would be the last one to accuse you of being lazy.

Thank you for posting the RPN solution. I'll definetly be enterng it and anaylzing and comparing it to the 71B vesion. Just to let you know, I do enter ALL your 71b programs and run them on my 71B.

"The 71B's BASIC code is just four lines, so very compact and extremely easy to grasp at a glance and understand what it is doing and how it does work"

I've never been a fan of "counting" lines of code as being a reference for whether its good code, easy to grasp, etc. Many times fewer lines just means more stuff on a single line - not necessarily easier to understand. The first thing I do with any program is rewrite it out with many lines in a structured format so that I can visually see the "for-next", "If-then" statement groups lined up. That's the only way I can visually see the program structure.

"Of course, out of sheer human laziness, one would prefer to directly get the solutions in RPN, RPL, or whatever, but be a sport and do some little work yourself, I'm already doing my part, don't you think ?"

I don't like to think of myself as being lazy, and maybe I don't post all my program results but I think you will find that I do "some work myself" and I have posted some of my results in the past and will be posting more results in the future.

"Remember, it's much easier to come up with some pretty reasonable excuses than to come up with some reasonably pretty solutions "

I totally agree. I don't think you'll ever find any of my solutions "Pretty" <g>.

Bill

12345

Re: S&SMC#9: My ** RPN ** solution & comments

Posted by ***Valentin Albillo*** *on* ***25 May 2005, 11:21 a.m.****, in response to Re: S&SMC#9: My ** RPN ** solution & comments, posted by Bill (Smithville, NJ) on 25 May 2005, 10:00 a.m.*

Hi, Bill:

Bill posted:

"With the amount of time you spend posting, replying to posts, creating challenges, articles for Datafile, and I assume you also have a job that requires your time [...]"

> Yes, I'm working right now in four commercial projects at a time, namely an VB 6 project, an VB.NET project, a Netbeans project and, last but not least, a Struts project. Not to mention I also have a wife and a pre-teen daughter to attend to, plus I'm finishing three Datafile articles and a couple of private programming projects ... and Yet-Another-S&SM Challenge ! :-)

"Thank you for posting the RPN solution. I'll definetly be enterng it and anaylzing and comparing it to the 71B vesion."

> You're welcome. And as I said, they're identical, it's a straightforward translation. BASIC, even when using recursion, translates fairly easily to RPN and vice versa.

"Just to let you know, I do enter ALL your 71b programs and run them on my 71B."

> May I suggest you enter them in Emu71 instead. They'll run orders of magnitude faster, you'll see all results past and present plus full listings in the screen at the same time, and you can copy & paste my listings instead of keying them in, saving time and laborious-to-find errors.
>
> And, by the way, do not miss my very next 71B article in Datafile, due next month. I guarantee it's awesome, and I mean it ! :-)

"Many times fewer lines just means more stuff on a single line - not necessarily easier to understand."

> I think I already made it clear why I do include as many statements as possible in a single line: to save precious Datafile's space.

"The first thing I do with any program is rewrite it out with many lines in a structured format so that I can visually see the "for-next", "If-then" statement groups lined up. That's the only way I can visually see the program structure."

> I can visualize most any program, whether 'pretty formatted' or not, specially such small routines. Matter of fact, I do tend to prefer them in a compact way, without having to move the eyes down the page to try and find the matching NEXT, say.

"I don't like to think of myself as being lazy"

> No offence meant, bad wording on my part. I wasn't implying you personally, it was a general statement. Also, I forgot to plant a smiley at the end.

"I don't think you'll ever find any of my solutions "Pretty""

> Who knows ? I've rather have an allegedly non-pretty solution than none at all. Post them and we'll see ...

Best regards from V.

Edited: 25 May 2005, 11:58 a.m.

Re: S&SMC#9-Question to Valentin

*Posted by **chris dean** on **24 May 2005, 3:22 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by Bill (Smithville, NJ) on 23 May 2005, 7:32 a.m.*

Surely the object of the challenge is to get the solution and using a 'calculator' style solution is a makes it all the more interesting. If the solver wants to use a PC then they are the losers.

Re: S&SMC#9-Question to Valentin

*Posted by **Thomas Okken** on **24 May 2005, 4:58 a.m.**, in response to Re: S&SMC#9-Question to Valentin, posted by chris dean on 24 May 2005, 3:22 a.m.*

Surely the object of the challenge is to get the solution and using a 'calculator' style solution is a makes it all the more interesting.

Couldn't agree more. I guess I'm in the "HP-71 is not a calculator" camp - for me, my gut feeling is that a calculator is a small handheld device, with little memory and speed, and which is programmed using keystroke programming. And programming such devices has a certain charm that BASIC just doesn't have -- a bit like using assembly language instead of C, very retro, and maybe just a little masochistic. ;-)

No disrespect to BASIC-lovers, of course! To each their own.

- Thomas

Edited: 24 May 2005, 4:59 a.m.

S&SMC #9, Act II

*Posted by **J-F Garnier** on **26 May 2005, 7:26 a.m.***

Valentin wrote:
" ... the number of solutions seems to grow forever that's not the case. Actually, 2492 is the maximum number of solutions for any N (in this case, for N=9 and N=10). After that (N=11, 12, etc), the number of solutions decreases steadily till N=25, which again has a unique, 25-digit solution".

My personal challenge was then to find this unique solution on the HP-71B (actually using Emu71). I wrote a program (non-recursive, with no GOSUB) that solves the problem in less that 40 seconds with Emu71 in Fast mode. I will post my solution in a few days to let you play with this challenge, if you like it.

J-F

Note: Emu71 Fast mode is just Emu71 with /fx option,
e.g. "Emu71 /f50" with a 1.7GHz processor, or even "Emu71 /f80" with a 2.3GHz processor (close to a virtual Saturn running at 80MHz...).

Emu71 /f50 ?

*Posted by **Valentin Albillo** on **26 May 2005, 7:32 a.m.**, in response to S&SMC #9, Act II, posted by J-F Garnier on 26 May 2005, 7:26 a.m.*

Hi, Jean-François:

J-F posted:

"Note: Emu71 Fast mode is just Emu71 with /fx option, e.g. "Emu71 /f50" with a 1.7GHz processor, or even "Emu71 /f80" with a 2.3GHz processor (close to a virtual Saturn running at 80MHz...)."

I didn't know that you could use a number after the "/f", I was using just "Emu71 /f", per se, no number after.

What does this number mean and how can you deduce its optimum value for a given processor and/or processor speed ?

As for your "personal challenge", that was my intention when mentioning that 25-digits had just an unique solution, that someone would "take the bait" and would decide to try and find that solution. I'm glad that you did, and even more that you used Emu71 to do it, with such good timing.

Best regards from V.

Emu71 Fast Mode - explaination

*Posted by **J-F Garnier** on **26 May 2005, 8:02 a.m.**, in response to Emu71 /f50 ?, posted by Valentin Albillo on 26 May 2005, 7:32 a.m.*

The number after the /f option set the size of the emulated opcode 'burst' before checking keyboard, updating display and timers and so on. As Emu71 is basically a DOS program, these calls to the OS I/O primitives also set the relative CPU usage given by the Windows OS due to the automatic idle detection. Max value is 255, but if you set a too high value, Emu71 realtime clock will not be emulated correctly.

There is no direct relation with the equivalent Saturn frequency, I said that /f80 was close to a 80MHz Saturn, but it's only by chance.

A convenient way to optimize the parameter for a particular machine is to use the CPU usage display (in the Task Manager utility), and to target about 80% CPU usage.

J-F

Valentin: you may notice that the speed-up is not effective in your last challenge solution. This is due to the CALL function, I don't know why right now and I'm working to correct this. That's why I don't use recursive calls in Emu71 for 'force brute' approach. BTW such challenges are very usefull to test emulator behaviors...

Re: Emu71 Fast Mode - explaination

*Posted by **J-F Garnier** on **26 May 2005, 8:44 a.m.**, in response to Emu71 Fast Mode - explaination, posted by J-F Garnier on 26 May 2005, 8:02 a.m.*

Correction:

With /f50 on a 1.7GHz processor the speed-up is actually about x140, measured by the following test program:

```
240 SUB SPEED
250 ! test emulator speed
260 ! a HP71B needs about 20 s for 2000 iterations
270 N=2000
275 T=TIME
280 FOR I=1 TO N @ NEXT I
290 T=TIME-T @ IF T<2 THEN N=N*2 @ GOTO 275
295 DISP T;"s. SPEED=";IP(N/T)/100;"TIMES HP-71B SPEED"
300 END SUB
```

J-F

Re: Emu71 Fast Mode - explaination

Posted by ***Valentin Albillo*** *on* ***26 May 2005, 9:05 a.m.****, in response to Re: Emu71 Fast Mode - explaination, posted by J-F Garnier on 26 May 2005, 8:44 a.m.*

Thanks a lot for the detailed explanation, Jean-François ! :-)

I'm truly glad that you find my challenges useful and interesting. Certainly, were it not for your Emu71 and kind answers to my requests, many of my 71B-related articles and challenges would've never seen the light.

Being able to use Emu71 for development and testing purposes increases productivity a hundredfold, which, with the very little free time I can gather for these activities, is absolutely of the essence.

Best regards from V.

S&SMC #9, Act II - Solution and RPL Programming Challenge

*Posted by **J-F Garnier** on **29 May 2005, 3:03 p.m.**, in response to S&SMC #9, Act II, posted by J-F Garnier on 26 May 2005, 7:26 a.m.*

Below is my solution. First, I noticed that to be divisible by 25, a number should end with 00,25,50 or 75 so it is enough to find all the 24-digit solutions. This means that double length arithmeric can be used on Saturn (and Capricorn...) machines (2*12 digits), with only the implementation of MOD, *10 and div10 operations, which is quite easy to do.

```
10 ! S&SMC #9, Act II
20 ! JFG, May 2005
30 H=24 ! # of digits (24 max)
40 C=0 ! solution counter
50 FOR I=10 TO 98 STEP 2 ! loop for 1st and 2nd digit positions
60 L=3 ! start tests from 3rd position
70 M=I*10 ! lower part of candidate number
80 N=0 ! higher part
90 'A': X=MOD(N*1.E+12,L)+MOD(M,L) @ IF X>=L THEN X=X-L ! x= [n,m] mod L
100 IF X<>0 THEN X=L-X ! find Lth digit
110 IF X>9 THEN 'D' ! is it a possible candidate?
120 M=M+X ! yes, built it.
130 IF L=H THEN C=C+1 @ DISP N;M @ GOTO 'C' ! found one!
140 'B': N=N*10+M DIV 100000000000 @ M=MOD(M,100000000000)*10 ! go on
[n,m]*=10
150 L=L+1 @ GOTO 'A'
160 'C': X=X+L @ IF X<=9 THEN M=M+L @ GOTO 'B' ! next try, if possible
170 'D': M=MOD(N,10)*100000000000+M DIV 10 @ N=N DIV 10 ! else [n,m]/=10
180 X=MOD(M,10)
190 L=L-1 @ IF L>2 THEN 'C'
200 NEXT I
```

The only three 24-digit solutions are:

```
 144408645048  225636603816
 360852885036  840078603672
 402852168072  900828009216
```

So the unique 25-digit solution is: 3608528850368400786036725.

You can notice that I used a few GOTOs and 4 labels. Actually, I'm a supporter of structured programming style with WHILE/LOOP/REPEAT/etc structures (such as in the HP71 JPCROM), so it can be a challenge to rewrite the program with no explicit GOTO.

Will someone implement this simple but non-trivial algorithm on HP48's RPL for instance?

J-F

Re: S&SMC #9, Act II - Solution and RPL Programming Challenge

*Posted by **Raymond Del Tondo** on **29 May 2005, 9:59 p.m.**, in response to S&SMC #9, Act II - Solution and RPL Programming Challenge, posted by J-F Garnier on 29 May 2005, 3:03 p.m.*

Hi J-F,

nice exercise.

For an HP-48 (or generally RPL) solution
I think the runtime structure will have to be rewritten,
as you do jumps into and out of 'loops' in your BASIC program.

My first steps would be maybe reordering
the test logic to omit labels 'C' and 'D',
and in consequence omitting label 'B'.

The following code leaves out some assignments,
and maybe has still some logic error inside,
but it is intended only for showing the principle
how it could be solved completely w/o GOTO;-)

However, it's deep in the night and I'll go to sleep soon...

Regards

Raymond

```
::  %24 'H' STO %0 'C' STO
    NINEYEIGHT TEN
    DO
    ::  %3 'L' STO INDEX@ #10* UNCOERCE 'M' STO %0 'N' STO

* 'A'-Loop:
            :: BEGIN
                ::
* some code
                    X %9 %> ITE
                    ::  (this seco was label 'D')
                        L %2 %> (leaves flag on stack for 'C')
                        TRUE    (leaves flag on stack for 'A' test)
                    ;
                    ::  M X %+ 'M' STO
                        L H %= NOTcaseTRUE     ('GOTO C')

                        C %1+ 'C' STO
                        FALSE                  (SKIP C)
                        FALSE   (leaves flag on stack for 'A' test)
                    ;

                    caseTRUE   (leaves flag on stack for 'A': Exit 'A')

                    ITE
                    ::  (this seco was label 'C')
                        X %9 %> (leaves flag on stack for 'B')
                    ;
                    TRUE
                    ?SKIP
                    ::  (this seco was label 'B')
                        L %1+ 'L' STO
                    ;

                    FALSE
                ;
                UNTIL
            ;

            INDEX@ #1+ INDEXSTO
        ;
    LOOP
;
```

Re: S&SMC #9, Act II - Solution with JPCROM

*Posted by **J-F Garnier** on **31 May 2005, 11:39 a.m.**, in response to S&SMC #9, Act II - Solution and RPL Programming Challenge, posted by J-F Garnier on 29 May 2005, 3:03 p.m.*

Here is my structured programming style solution, using the HP-71B JPCROM:

```
  10 ! S&SMC #9, Act II
  20 ! JFG, May 2005 - with JPCROM
  30 H=24 ! # of digits (24 max)
  40 C=0 ! solution counter
  50 FOR I=10 TO 98 STEP 2  ! loop for 1st and 2nd digit positions
  60   L=2 ! start of search
  70   M=I ! lower part of candidate number
  80   N=0 ! higher part
  90   REPEAT
 100     REPEAT
 110       N=N*10+M DIV 100000000000 @ M=MOD(M,100000000000)*10 ! go on
[n,m]*=10
 120       L=L+1
 130       X=MOD(N*1.E+12,L)+MOD(M,L) @ IF X>=L THEN X=X-L ! x= [n,m] mod L
 140       IF X<>0 THEN X=L-X ! find Lth digit
 150       IF X<=9 THEN M=M+X ! if possible candidate, build number
 160     UNTIL X>9 OR L=H ! until no more possible, or solution reached
 170     IF X<=9 AND L=H THEN C=C+1 @ DISP N;M ! found one!
 180     REPEAT
 190       M=MOD(N,10)*100000000000+M DIV 10 @ N=N DIV 10 ! go back [n,m]/=10
 200       X=MOD(M,10)
 210       L=L-1
 220       X=X+L ! next try
 230     UNTIL X<=9 OR L=2 ! until a possible try, or end of search
 240     M=M+L
 250   UNTIL L=2 ! until end of search
 260 NEXT I
```

The algorithm is now much easier to read and understand, at the expense of some reduntant tests.

Nice exercice, indeed.

J-F

Yep, the JPC rom is a jewel - every 71b owner should have it!

Posted by ***Gene*** *on* ***31 May 2005, 12:01 p.m.****, in response to Re: S&SMC #9, Act II - Solution with JPCROM, posted by J-F Garnier on 31 May 2005, 11:39 a.m.*

Wonderful lex file.

Gene

Re: S&SMC #9, Act II - Solution with JPCROM

Posted by ***Valentin Albillo*** *on* ***31 May 2005, 12:13 p.m.****, in response to Re: S&SMC #9, Act II - Solution with JPCROM, posted by J-F Garnier on 31 May 2005, 11:39 a.m.*

Hi, J-F:

Excellent solution to the extended version of my S&SMC#9. Frankly, I didn't expect anyone at all to come up with the unique 25-digit solution but then I wasn't expecting you to try your hand at it, either ! :-)

By the way, despite your titanic effort the topic isn't exhausted yet. What about bases other than 10 ? ;-)

Glad that you enjoyed my challenge, hope to see you attempt the next one as well.

Best regards from V.

Re: S&SMC #9, Act II - Solution with JPCROM

*Posted by **Raymond Del Tondo** on **31 May 2005, 1:22 p.m.**, in response to Re: S&SMC #9, Act II - Solution with JPCROM, posted by J-F Garnier on 31 May 2005, 11:39 a.m.*

Hi J-F,

did you see my last post?

Your today's solution will be even easier to port to RPL;-)

Regards

Raymond

Re: S&SMC #9, Act II - Solution with JPCROM

*Posted by **J-F Garnier** on **31 May 2005, 3:20 p.m.**, in response to Re: S&SMC #9, Act II - Solution with JPCROM, posted by Raymond Del Tondo on 31 May 2005, 1:22 p.m.*

Hi Raymond,

Yes I read your post, this was one reason to me to write a GOTO-less version (the other reason was to demonstrate the JPCROM powerfull features).

But User RPL is good enough for this problem :-)

J-F

Edited: 31 May 2005, 3:25 p.m.

Short & Sweet Math Challenge #10: Counting beans

*Posted by **Valentin Albillo** on **2 June 2005, 12:37 p.m.***

Hi all,

This new **S&SMC #10** completes the "trilogy" of *find-the-number* style challenges, and once again, we're looking for a 10-digit integer number (what else ?) and the answer is unique. So grab your favorite HP (preferably) calculator/handheld and do write a program for it to try and solve

The challenge

Find a 10-digit integer number such that its 1st digit counts the number of 0's in the whole number itself, its 2nd digit counts the number of 1's, and so on till its 10th digit, which counts the number of 9's present. For example, this would be a solution:

```
        3541500926
```

were it not for the fact that there are two "0", not three; there's a single "1", not five, there's a single "2", not four; etc, etc.

The solution is **unique**. Numbers beginning by one or more "0"'s are *not* 10-digit numbers. As usual, you're expected to produce a program for your chosen calculator(s) that upon running will find the one and only solution. There's no need to let it run till it makes sure there are no others, but if you want to, so much the better.

Midway next week I'll post my solution (plus comments), which is a simple 5-line program for the HP-71B which finds the only solution in an average time of 15 seconds (nearly instantaneous under Emu71). It uses very, very little programmer's insight (see Recommended Guidelines, below), just the bare minimum and obvious heuristic to speed up the search. I may also provide an HP42S and/or HP-41C and/or HP-15C solution if available time permits. You'll have the whole weekend and then some to try your hand at it.

Recommended Guidelines

:

- Absolutely refrain from using a PC, laptop, or PDA (unless running some HP calc emulator/simulator) to solve the challenge. A Mathematica, Visual Basic, C++, or Java solution is *useless* to the intended purporses of this challenge, in fact actually *defeats* them and is to be considered unpolite on purpose.

- Refrain from using your own smarts to solve the challenge, then perpetrate a program which incorporates so much of your previous study of the problem that it has pretty little to do. In an extreme case, you find the solution entirely by hand, then your program simply prints it out !.

  On a more typical example, you let the puzzle in so solved an state that the program finds it only too easy to deal with the pitiful remains. That's *not* it. The idea is that **the machine** does most of the work for you, **not** the other way around.

So keep your heuristics *simple* and show instead your programming muscle (if any), not your own puzzle-solving abilities. Try to consider yourself a *puzzle-solver dummy* but an HP-calc *programmer genius*, and see what comes out. Or else.

Best regards from V.

Edited to correct a major typo, doesn't affect any attempts to solve it

Edited: 3 June 2005, 8:54 a.m. after one or more responses were posted

Re: Short & Sweet Math Challenge #10: Counting beans

Posted by ***.*** *on* ***3 June 2005, 2:44 a.m.****, in response to Short & Sweet Math Challenge #10: Counting beans, posted by Valentin Albillo on 2 June 2005, 12:37 p.m.*

Hi,

This is an interesting challenge. How do you keep coming up with them?

> Quote:
>
> ---
>
> It uses very, very little programmer's insight (see Recommended Guidelines, below), just the bare minimum and obvious heuristic to speed up the search
>
> ---

I've written a program that gives the correct answer, but it takes far too long. It simply starts at zero and checks every possible number. I'm probably missing the obvious but I can't think of any way to speed up the result right now *g*

Re: Short & Sweet Math Challenge #10: Counting beans

Posted by ***Arnaud Amiel*** *on* ***3 June 2005, 3:30 a.m.****, in response to Short & Sweet Math Challenge #10: Counting beans, posted by Valentin Albillo on 2 June 2005, 12:37 p.m.*

This one looks easy and I could solve it in my head without paper in about 5mn so I though I will give it a try on my favourite the hp55. Unfortunately the lack of indirect register adressing and the few available steps make it quite hard. I will continue trying though.

My algorithm: Assume there are lots of zeros so start with 9000000000 And correct your number one step at a time:

```
9000000001
9100000001
8100000010
8200000010
...
```

You quickly get to the solution (3 or four more steps)

Arnaud

Edited: 3 June 2005, 7:08 a.m. after one or more responses were posted

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Chris Dean** on **3 June 2005, 5:17 a.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Arnaud Amiel on 3 June 2005, 3:30 a.m.*

Numbers of this type are called autobiographical numbers.

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **J-F Garnier** on **3 June 2005, 9:14 a.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Arnaud Amiel on 3 June 2005, 3:30 a.m.*

I found a solution based on the same principle. This is an iterative algorithm that converge to the solution very quickly. It just counts the occurrence of each digit, and continuously update the digits until a stable state is found. But of course, it doesn't prove that the solution is unique.

I'm not sure that this is the kind of solution that Valentin is asking for, but I find it interesting.

J-F

```
  10 ! S&SMC #10 JFG June 2005
  20 OPTION BASE 0
  30 DESTROY D
  40 DIM D(9) ! the 10 digits
  50 K=0 ! iteration counter
  60 REPEAT
  70   F=0
  80   FOR I=0 TO 9 ! for each position
  90     N=0
 100     FOR J=0 TO 9 ! scan the present values
 110       IF D(J)=I THEN N=N+1
 120     NEXT J
 130     IF N>=10 THEN N=9 ! in case all digits are the same
 140     IF N<>D(I) THEN D(I)=N @ F=1 ! and update the position if needed
 150   NEXT I
 160   K=K+1
 170   DISP K;':'; @ FOR I=0 TO 9 @ DISP STR$(D(I)); @ NEXT I @ DISP
 180 UNTIL F=0  ! until no more change
```

(Replace the REPEAT/UNTIL loop with a GOTO if you don't use the JPC Rom)

Edited: 3 June 2005, 9:16 a.m.

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Thibaut** on **3 June 2005, 10:53 a.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Arnaud Amiel on 3 June 2005, 3:30 a.m.*

In order to check my algorithm, i did in excel first.

BUT : it doesn't work with all numbers : if the sequence 7110000100 is reached, it generates 6300000100 wich loops back to 7110000100. (it does work with 6548456111 for example)

Arnaud, I looked at your iteration and I'm under the impression that you apply "too fast" the calculation. My sequence starting with 9000000000 (I had the same idea as yours) is :

9000000000 9000000001 8100000001 7200000010 7110000100 6300000100 7101001000 6300000100

I've posted the excel file here

Your feedback is much appreciated !

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Arnaud Amiel** on **3 June 2005, 2:14 p.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Thibaut on 3 June 2005, 10:53 a.m.*

I was worried that there could be some numbers that wouldn't simplify but hadn't looked into it enough. I thought any input would converge towards the solution but it is clearly not the case. As my plans were to use a hp55, I would have actually started with 0000000000 which would have converged.

As to the simplification algorithm, you are also right, what I wrote does not make sense. I am still working on it but I hope to have something slightly faster (number of iterations) than what you wrote.

I also believe that I will give up on the 55 and go to the 49 (I don't have a 65 and my 97 is currently in pieces)

Thanks for pointing out that the algorithm is not always converging, my intuition was wrong so the input has to be chosen carefully.

Arnaud

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Arnaud Amiel** on **3 June 2005, 6:21 p.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Arnaud Amiel on 3 June 2005, 2:14 p.m.*

Still no answer for the hp55 but I tried my algorithm on the 49g+ and I get the right number in 5.56 seconds. The program will run on a 48G and with very little changes on an S(X) or a 28 (The REVLIST command is used to nicely display the final result)

Here is the RPL code:

```
<<
[ 0 0 0 0 0 0 0 0 0 0 ] -> N   @we work on 10 digits array
@Thibaut noted above that the starting value may be important. His example
does not work here though
<<
DO
  N @to compare before and after processing
  0 9 FOR I
          N I DUP2
          0 -> V C  @The Value we check, the Counter of digits
          <<  OBJ-> OBJ-> DROP @We get all the digits on the stack
          1 SWAP FOR A
                     IF V == THEN 'C' INCR DROP END @We count for the Value
          NEXT
          C @Number of digits with the Value
          >>
          SWAP 1 + SWAP PUT 'N' STO @We update the table
  NEXT
UNTIL N SAME END @We have to the solution
N
>>
@Next we transform the array into a real
OBJ-> OBJ-> DROP ->LIST REVLIST OBJ-> DROP
1 9 FOR I 10 * + NEXT
>>
```

I know some people around here think RPL does not look nice but it does the job...

Arnaud

Edited: 4 June 2005, 2:27 a.m.

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **chris dean** on **3 June 2005, 2:39 p.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Thibaut on 3 June 2005, 10:53 a.m.*

Thibaut I think you answer is incorrect. It represents a number with 6 zeros, 3 ones and 1 seven which it is not.

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Thibaut** on **4 June 2005, 3:21 p.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by chris dean on 3 June 2005, 2:39 p.m.*

That's not my answer. Some "root" numbers generate the correct number. Simply open my excel file and you'll het the right answer. Others don't, therefore I wonder where my algorithm is wrong or to say this differently, whith what kind of numbers it works and what kind of numbers it doesn't.

Valentin, your support is much appreciated !

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Chris Dean** on **5 June 2005, 4:36 a.m.**, in response to Re: Short & Sweet Math Challenge #10: Counting beans, posted by Thibaut on 4 June 2005, 3:21 p.m.*

```
Dim A(9) As Integer
Dim B(9) As Integer
Dim iRow As Integer
Dim strOut As String

Dim iCount As Integer
Dim jCount As Integer
Dim iFlag As Integer

A(0) = 9

iFlag = 1
While iFlag = 1
    iFlag = 0
    For iCount = 0 To 9
        B(iCount) = 0
        For jCount = 0 To 9
            If A(jCount) = iCount Then
                B(iCount) = B(iCount) + 1
            End If
        Next
        If A(iCount) <> B(iCount) Then
            A(iCount) = B(iCount)
            iFlag = 1
        End If
    Next
    strOut = ""
    For iCount = 0 To 9
        strOut = strOut + Trim(Str(A(iCount)))
    Next
    Print strOut
Wend
```

The algorithm above does not oscillate and reaches the solution in about 4 or 5 prints. The reason it does not oscillate is the line where A(iCount) is set to B(iCount). This can be applied to your algorithm and hopefully solves your problem.

I found programming this on my HP49G+ and Hp17BII+ too stressful and after developing the algorithm in Excel modified it to run on an old Casio FX-700P which took ages to run! I really need an HP71!

Regards

Chris 123

Re: Short & Sweet Math Challenge #10: Counting beans

*Posted by **Bram** on **3 June 2005, 8:17 a.m.**, in response to Short & Sweet Math Challenge #10: Counting beans, posted by Valentin Albillo on 2 June 2005, 12:37 p.m.*

Hi Valentin,

I found the answer in less than a couple of minutes by hand, but, as you stated, the challenge is in the programming, so I took my HP-32SII and voilá.
A very brute force program that will find the answer after some ages of running. It tests a number in about 3 seconds, so.....
I cannot yet possibly think of a way to have it run faster. (but *that* wasn't the challenge ;-)

```
999999999
STO X
LBL A
1
STO+ X
10
STO i
0
LBL Z   ; Zero reg A .. reg J
STO(i)
DSE i
GTO Z
1.010   ; do ten times
STO T
RCL X
PSE
1E9
/
STO Z
LBL S   ; Split number into digits and count them
RCL Z
FP
STO Z
LASTx
IP      ; one single digit
STO i   ; save it as an index
ISG i   ; 0..9 becomes 1..10 (regs A to J)
ABS     ; dummy instruction
ISG(i)  ; increment occurance of this digit
ABS     ; dummy instruction
1E10
/
STO+ Z
10
STO* Z  ; digit has been added to the tail of the number
ISG T
GTO S
1.010   ; do ten times
STO i
LBL C   ; Compare each digit with occurance
RCL Z
FP
STO Z
LASTx
IP      ; one single digit
STO T   ; save it
1E10
/
```

```
STO+ Z
10
STO* Z  ; digit has been added to the tail of the number
RCL(i)  ; number of occurances of 'ranked value'
RCL T   ; position of digit, the 'ranked value'
x<>y?
GTO A
ISG i
GTO C
RCL X
```

thank you

Re: Short & Sweet Math Challenge #10: For the 15C

*Posted by **Arnaud Amiel** on **5 June 2005, 2:44 p.m.**, in response to Short & Sweet Math Challenge #10: Counting beans, posted by Valentin Albillo on 2 June 2005, 12:37 p.m.*

Having given up on using the 55, I still wanted to try it in RPN. This is the same algorithm as I posted for the 49. The following program was written for the 15C on which it takes about 14mn to complete. It should be easy to port it to some other calcs with a dozen of inderectly addressable registers.

```
This just loops through B until convergence:

LBL A
CLEAR REG
GSB C
STO .3
LBL 0
GSB B
GSB C
RCL .3
TEST 5     ?X=Y
RTN
R|
STO .3
GTO 0

This tidy the content of registerS 0-9

LBL B
.00901
STO .1
LBL 1
0
STO .0
.00901
STO .2
LBL 2
RCL .2
STO I
RCL .1
INT
RCL (i)
TEST 6     ?X#Y
GTO 3
1
STO+ .0
LBL 3
ISG .2
GTO 2
RCL .1
STO I
RCL .0
STO (i)
ISG .1
GTO 1
RTN

This takes the content of re 0-9 and returns a number
LBL C
.00901
STO .1
```

```
0
LBL 4
RCL .1
STO I
R|
10
*
RCL (i)
+
ISG .1
GTO 4
RTN
```

I hope there is no typo...

Arnaud

Re: Short & Sweet Math Challenge #10: For the 15C

*Posted by **Valentin Albillo** on **6 June 2005, 9:38 a.m.**, in response to Re: Short & Sweet Math Challenge #10: For the 15C, posted by Arnaud Amiel on 5 June 2005, 2:44 p.m.*

Hi, Arnaud:

Thanks for your interest in S&SMC#10, a few comments on your program for the HP-15C:

- You do use the constant ".00901" in several places. First of all, you don't need the "01" part, as all HP models implementing ISG and DSE consider the default increment to be 1 when not specified, so your constant need be just ".009", which is two steps shorter and faster.
- Also, as you're using it in several places, it will save program steps and be faster too if you store said constant in a register at the beginning of your program, then recall it as needed. As the constant is between 0 and 1, an ideal place is in the #RAN register, so simply
- `           .009`
- `           STO #RAN`
- `            ...`
- `           RCL #RAN`
- `           STO .1`

  would save a lot of bytes and run faster, too, while still not using any numbered register.
- Remember the HP-15C does have RCL arithmetic, so steps such as
- `           RCL (i)`
- `             +`

  can be shortened to RCL+ (i), again saving steps and running time (as well as one stack level). Also, you don't need a final RTN, end-of-program-memory acts as a default RTN if encountered during program execution.
- You do make use of "." registers. Your program would be shorter and faster if you rearranged your logic to use .0 to .9 indirectly, saving 0-9 for direct operations instead. For instance, your sequence
- `        1`
- `        STO+ .0`
- `        LBL 3`

  would then become a single ISG 0 instruction.
- You've arranged your logic to count 'upwards', which requires ISG and the ".009" constant. It would save program steps and execution time if you rearranged it to count 'downwards', changing that .009 to 9 and those ISG to DSE.
- The routine at LBL B is called just once (GSB B). You would save three steps (GSB, LBL, RTN) and a lot of time (as label search is so slow) by simply inserting its steps directly at the location of the call.

"I hope there is no typo..."

The best way to ensure this is simply to re-enter the program in your calculator from your own post. If it runs ...

Thanks for your contribution and

Best regards from V.

Re: Short & Sweet Math Challenge #10: For the 15C

*Posted by **Arnaud Amiel** on **6 June 2005, 11:55 p.m.**, in response to Re: Short & Sweet Math Challenge #10: For the 15C, posted by Valentin Albillo on 6 June 2005, 9:38 a.m.*

Thanks a lot for your comments. I am not really keen on rpn and this was my first program that I could call slghtly "evolved". I am much more comfortable with RPL although I am not well acquainted with all that was introduced in the 49...

Once again thanks a lot. And I am looking forward to your 15C solution.

Arnaud

S&SMC#10: Final Notice

*Posted by **Valentin Albillo** on **6 June 2005, 10:14 a.m.***

Hi all:

Many thanks to those of you interested in my S&SMC#10: Counting Beans challenge, most specially to those who found the time to post a contribution, all submitted are very clever and interesting.

Tomorrow I'll post my original solution in two flavors, a 5-line HP-71B BASIC program (can be reduced to 4 lines) which finds the only solution in 15 sec. average (nearly inmensurable in Emu71), and its optimized RPN translation for the HP-15C, which is a fairly small, 52-step program (can be reduced to 51 steps) which finds the solution in 2 min or so. I'll include relevant comments as well.

So, for those of you still wanting to contribute your own approach before I publish mine(s), you have an additional day to do so. Mind you, you can post them afterwards and I'll be equally pleased as well, of course.

Best regards from V.

Re: S&SMC#10: Final Notice

*Posted by **Giancarlo** on **6 June 2005, 4:25 p.m.**, in response to S&SMC#10: Final Notice, posted by Valentin Albillo on 6 June 2005, 10:14 a.m.*

Hello Valentin. Your S&SMCs are very interesting, even if I haven't got any time to solve anyone of them, but..... never thought to collect all of them in a document, with your "official" solutions and the best ones from other contributors? If you already did that, sorry for my useless suggestion - where could it be possible to find such document? Thank you. Cheers. Giancarlo

---

Re: S&SMC#10: Final Notice

*Posted by **Valentin Albillo** on **7 June 2005, 4:44 a.m.**, in response to Re: S&SMC#10: Final Notice, posted by Giancarlo on 6 June 2005, 4:25 p.m.*

Hi, Giancarlo:

It's a pretty good idea but regrettably no such 'compilation' does exist. It's all a matter of available free time and priorities. I'm already using lots of time to concoct these challenges and my solutions to them, plus all the postings, time which should really be allocated to other matters.

Making a compilation as you suggest would require even more time, if done properly and I simply can't afford it right now. Perhaps some other interested people are collecting them and can assemble and make available such a compilation.

Also, in the early stages, not all my challenges were succesful, I seem to remember that a couple of them gathered no useful responses at all, perhaps because people thought they were too complicated or deeply mathematical in nature.

Thanks for your interest and

Best regards from V.

S&SMC#10: My Original Solutions & Comments

*Posted by **Valentin Albillo** on **7 June 2005, 5:15 a.m.**, in response to S&SMC#10: Final Notice, posted by Valentin Albillo on 6 June 2005, 10:14 a.m.*

Hi all,

Thanks a lot to all of you interested in this small challenge of mine, we've got a surprising number of clever programs for a variety of HP machines, written in assorted languages including RPL, several RPN flavors, and even BASIC.

Here's my original RPN solution for the HP-15C, a small, 52-step program which uses matrix operations to find the solution quickly:

```
    01 *LBL A      17  GTO 8          32  CLX            47  RCL MATRIX A
    02   1         18  RCL MATRIX A   33  STO MATRIX C   48   -
    03  MATRIX 0   19 *LBL 3          34  RCL I          49  MATRIX 7
    04   10        20  GSB 5          35  STO 1          50  RETURN
    06  STO I      21  X=0 ?          36 *LBL 1          51 *LBL 0
    07  DIM A      22  GTO 0          37   1             52  RCL MATRIX C
    08  DIM B      23  RCL MATRIX C   38  RCL+ B
    09  DIM C      24  GSB 5          39  X<> 1
    10 *LBL 2      25  X=0 ?          40  ISG C
    11  MATRIX 1   26  GTO 2          41 *LBL 1
    12 *LBL 8      27  RCL MATRIX C   42  X<> 1
    13  RAN#       28  STO MATRIX A   43  DSE 1
    14  RCL* I     29  GTO 3          44  GTO 1
    15  INT        30 *LBL 5          45  RESULT B
    16 uSTO A      31  STO MATRIX B   46  RCL MATRIX C
```

```
    Note: Step 16 is a "User" STO operation, and must be entered while
temporarily
          in USER mode, a small "u" should appear next to the step number.
All other
          STO and RCL operations must be entered *out* of USER mode.
```

It doesn't use any allocatable numbered registers but the permanent ones, namely R_0-R_1 (system indexes for matrix operations) and R_I (index register used here merely to hold the constant 10).

The program requires extremely very little user-provided heuristics. Not even the fact that the digits must add up to 10 or that there can't be two 9's, say, is made use of. Instead, the only thing the program knows is that it must compute a function **f(N)**, say, than when applied to a 10-digit N returns another 10-digit result where each digit represents a count of the corresponding digit in the original number, so we have:

```
      f(N) = M
```

Obviously, the value of N we're looking for has the unique property that *when **f** is applied to it, it returns unchanged*, that is:

```
      f(N) = N
```

The program simply goes on to solve this equation. There are a number of reasonable ways we could proceed, but the most successful is to simply select an starting value of N, say N_0, then compute:

$f(N_0)$ -> N_1, $f(N_1)$ -> N_2, ...

till for some N_i we do have

$N_{i+1} = f(N_i) = N_i$

As f(N) maps 10-digit numbers into 10-digit numbers and there are only so many of them, we must have *cycles*, where successive applications of f(N) eventually repeat some previous value. We are interested in the *1-cycle*, where f(N) = N, but other cycles can and do exist.

To cater for this, my program initially selects a 10-digit value of N at random, then keeps on applying f(N) to the resulting values while checking if some cycle has been detected. If it is a 1-cycle, this is the solution and the program stops with the solution in the display. Else, if the program has stumbled into a greater order cycle, it simply restarts again, selecting another random starting value. Eventually, it succeeds, provably after very few restarts indeed, and the solution is displayed.

To maximize speed, my program uses a matrix representation for numbers, where a 1x10 matrix represents a 10-digit number. This way we can make use of the *powerful* and *fast* HP-15C matrix operations. Checking for repetition, for instance, is as simple as testing if the Row Norm (MATRIX 7) of the difference of two matrices is zero, which can be done in a few steps and with no slow user-code loops involved.

To run the program, follow these steps:

- Allocate enough registers to dimension the three 1x10 matrices needed (assuming this is the only program in program memory, 26 or any lower number will do):
- `26, DIM (i)`
  MEM should display [26 30 9-1] (i.e., 26 numbered registers plus R_0 and R_I + 30 registers reserved for matrices)
- Optionally, specify a seed for the random number generator. This step is not necessary, as the program will always converge from any starting seed, but if you want to repeat my timing, I always use PI in such cases:
- `PI, STO #RAN,`
- Now simply run the program:
- `GSB A -> (after 2 min 56 sec) -> C   1  10`
  The program stops displaying the matrix which represents and holds the solution found. You just need to display its elements as usual, for instance:
```
    MATRIX 1, USER,  RCL C -> (C1,1)  -> 6
                     RCL C -> (C1,2)  -> 2
                     RCL C -> (C1,3)  -> 1
                     RCL C -> (C1,4)  -> 0
                     RCL C -> (C1,5)  -> 0
                     RCL C -> (C1,6)  -> 0
                     RCL C -> (C1,7)  -> 1
                     RCL C -> (C1,8)  -> 0
                     RCL C -> (C1,9)  -> 0
                     RCL C -> (C1,10) -> 0
```
  which represents the unique solution: **6210001000**

The program can be made a step shorter using a flag, but I like it better as it is now. The equivalent version for the HP-71B, perhaps easier to understand and translate to other languages, is:

```
    10 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),X(9),T(9) @ RANDOMIZE
    20 FOR I=0 TO 9 @ N(I)=INT(RND*10) @ NEXT I
    30 MAT X=N @ GOSUB 50 @ IF NOT CNORM(X) THEN MAT DISP T @ END
    40 MAT X=T @ GOSUB 50 @ IF CNORM(X) THEN MAT N=T @ GOTO 30 ELSE 20
    50 MAT T=ZER @ FOR I=0 TO 9 @ T(X(I))=T(X(I))+1 @ NEXT I @ MAT X=T-N @
RETURN
```

which is 5 lines, 195 bytes long. A shorter version is possible, namely this 4-liner:

```
    1 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),X(9),T(9) @ RANDOMIZE
    2 FOR I=0 TO 9 @ N(I)=INT(RND*10) @ NEXT I @ MAT T=N
    3 GOSUB 4 @ IF I THEN MAT DISP T @ END ELSE GOSUB 4 @ MAT N=T @ IF I THEN
2 ELSE 3
```

```
    4 MAT X=T @ MAT T=ZER @ FOR I=0 TO 9 @ T(X(I))=T(X(I))+1 @ NEXT I @ MAT
X=T-N @ I=CNORM(X)=0 @ RETURN
```

but I think this is really overkill, besides it's actually one byte longer and slightly slower.

Some (long) statistical tests reveal that this kind of program always finds a solution after generating an average of 13.5 ten-digit numbers, though values as low as 1 (directly stumbling upon the solution by chance) or as high as 100 (entering n-cycles repeatedly) are possible, though with very low probabilities. Of course, even in the worst case of 100+ numbers generated and tested, an actual HP-71B can do it in just a few seconds and Emu71 takes usually much less than a second.

See you in S&SMC#11, thanks for your interest and

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

*Posted by **Chris dean** on **7 June 2005, 6:35 a.m.**, in response to S&SMC#10: My Original Solutions & Comments, posted by Valentin Albillo on 7 June 2005, 5:15 a.m.*

Valentin

I am sure everyone who took part in the challenge would like thank you for the time and effort you spent on them. Both in supplying them and in responding.

awaiting #11

Regards

Chris Dean

Thank you very much, Chris ! :-) [N.T.]

*Posted by **Valentin Albillo** on **7 June 2005, 6:59 a.m.**, in response to Re: S&SMC#10: My Original Solutions & Comments, posted by Chris dean on 7 June 2005, 6:35 a.m.*

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

Posted by ***Don Shepherd*** *on* ***7 June 2005, 9:00 p.m.****, in response to S&SMC#10: My Original Solutions & Comments, posted by Valentin Albillo on 7 June 2005, 5:15 a.m.*

Hi Valentin. Great problem, and I'm going to study your 15c solution in detail because I want to learn more about matrices.

Having just bought a 33s, I wanted to see how it's programming worked, so I attempted to code your challenge on it. My results are below.

I also started by introducing a random 10 digit number and calculating its digit distribution. Then I fed the results back into the algorithm two more times, giving me 3 numbers. If number 3 = number 2, I exit because that would be the magical number. If number 3 = number 1, then I have a pattern ABABAB, so I start again with a new random number. Otherwise I generate the next 3 numbers and test them. The final result is obtained after about 2 or 3 random numbers are used, in general (although it once took 9 random numbers). It is a fascinating problem and I'm going to study it some more.

Anyhow, thanks for the idea.

Don Shepherd

```
Register        Usage
A-J             count of num 0s, 1s, etc.
K               temp holds # being examined
L               constant 10
M               digit distr number
N               first iteration
O               second iteration
P               third iteration


LBL A           entry point
10
10^X            create 10 digit number
RAND            randomly
X
IP              take integer part of it
LBL B
XEQ C           create its digit distribution
29              store in stat register 29
STO i           because C routine clears vars
RDOWN
STO (i)
XEQ C           create dist based on first num
30              store in stat register 30
STO i
RDOWN
STO (i)
XEQ C           create dist based on second num
STO P           store directly in P
29              restore first two nums to N and O
STO i
RCL (i)
STO N
30
STO i
RCL (i)
STO O
```

```
RCL P
X=Y             if O = P, stop you are done
RTN
RCL N           if P = N, two are repeating
X=Y             so get new random number
GTO A
GTO B           get next three numbers
LBL C           given a #, it calcs digit dist
CLRVARS          A-M (stat regs not cleared)
STO K           number to create dist from
10
STO L           constant 10
RDOWN
LBL D           loop to get next digit
RCL L
RMDR
1
+               digit zero maps to Reg A
STO i
1
STO + (i)       increment digit count R(A-J)
RCL K
RCL L
INTGDIV          adjust number to get next digit
STO K
X<>0            loop until all digits mapped
GTO D
RCL L
STO i           start at register J (10) (ones position)
LBL E
RCL (i)          start at reg J and work down to A
RCL L
RCL i
IP
-
10^X            power of 10 multiplier
X
STO + M          add to reg M
DSE i           work way down to A
GTO E           loop
RCL M           the final digit dist
RTN
```

Re: S&SMC#10: My Original Solutions & Comments

*Posted by **Valentin Albillo** on **8 June 2005, 6:59 a.m.**, in response to Re: S&SMC#10: My Original Solutions & Comments, posted by Don Shepherd on 7 June 2005, 9:00 p.m.*

Hi, Don:

"Great problem, and I'm going to study your 15c solution in detail because I want to learn more about matrices."

Thanks a lot for your kind comments, and for the 33s program you posted. The algorithm you use is very similar to the one featured in my HP-15C solution. It's actually quite amazing that you managed to write such a program for your 33s right after purchasing it.

As for matrices, it's a real pity that the HP32S, SII, and 33S left out support for them. Matrix operations might seem very specialized at first, but once you understand how they work, they tend to be extremely useful almost for any application, frequently resulting in much more concise, faster programs.

Take this challenge, for example. Who could guest at first sight that using matrix operations would be advantageous ? You would think that a bunch of for-next loops would be needed, but matrices ? Yet representing the numbers as matrices brings the double advantage of being able to deal with their digist individually, as matrix elements, without wasting time extracting them from the original number, and also you can process the number as a whole, using matrix operations that affect the entire matrix.

The best of both worlds, once again. Thanks again for your interest and

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

*Posted by **Jeff O.** on **8 June 2005, 1:15 p.m.**, in response to S&SMC#10: My Original Solutions & Comments, posted by Valentin Albillo on 7 June 2005, 5:15 a.m.*

Valentin,
First, I'll second (or third, or fourth..) the thanks to you for the S&SM Challenges. While I don't always have the time or abilities (or both) to have a crack at them, they are always educational and entertaining.

I did manage to find the solution prior to reading your "Original Solutions & Comments" post above. No clever tricks using matrix capabilities, no knowledge that a recursive approach would lead to a solution. Just brute force; pick a number, test to see if it is the answer, if not, increment the previous guess and check that and so on. I relied on the speed of Free42 to insure a quick solution. My initial simple heuristic was only that the answer must be greater than 1,200,000,000 since anything less than that would start 1,1…, which is logically inconsistent since it has two ones. After determining that my first program based on the above would take about a week to get the answer, I thought about the problem some more and added additional constraints, such as the sum of the digits must equal 10, there can be no nines, eights or sevens anywhere in the answer (which of course means that the answer must end in 000, allowing me to increment my guesses by 1000 rather than by 1), I came up with a program that finds the solution in just under 2 minutes. Again, that's using Free42 on my PC; a real 42S would take ***_much_*** longer (about 3 months according to a quick check). In case anyone is interested, the program listing is below. (On a side note, I guess I did apply too many heuristics to the problem. As I was attempting to determine if sixes could also be eliminated from consideration, I stumbled upon the solution. Sheer luck, I am sure.)

Of course your 15C solution is very clever, capitalizing both on the capabilities of the 15C, and on mathematical knowledge about the problem. With time, I think I can probably figure out the workings of the 15C program. I may attempt to use matrix processing capabilities of the 42S to facilitate crunching the numbers to implement the process that I used to find the answer. However, I'm not sure that I ever would have or could have determined that a recursive procedure in which the initial guesses and new guesses are randomly generated would have ever led to a solution. This leads me to a question. Is the procedure that you implemented mathematically guaranteed to lead to a solution? I guess it seems to me that the procedure depends to some extent on the sequence of numbers generated by the (pseudo) random number generator on the calculator. If an initial guess that is not in the sequence that leads to the solution is never tried, then the solution will not be found, will it? I guess I worry that I might try an initial seed that would result in a lot of "tedious mucking about"[1] in number sequences that would not lead to the solution. Therefore, my tendency would be to start at 0000000001, and count up. This could take a long time, depending on the lowest number that happens to reside in a (or is it the?) sequence that leads to the solution.

Thanks again and best regards.

HP42S (and Free42) program listing:

```
00 { 165-Byte Prgm }
01>LBL "FIND2"
02 RCL "LB"     :recall the current number
03 1E3          :steps 3 and 4 strip off the last three digits since they are
04 ÷            : zero, i.e., no sevens, eights or nines
05 STO "NUM"    :store it in a working register
```

```
06 CLRG         :initialize registers that will store and count digits
07 3            :steps 7 and 8 initialize R11 (that will sum the number of
08 STO+ 11      : zeroes) to three to account for the last three digits.
09 7
10 STO "A"      :initialize counter to loop through the 7 remaining digits
11>LBL 01       :routine to break the number into digits and count them
12 RCL "NUM"
13 RCL "NUM"
14 10
15 MOD          :strip the number in the rightmost position
16 7
17 X<=Y?        :if the number is 7, 8 or 9, branch out of loop to get the
next
18 GTO 03       : 10 digit number to check
19 Rv           :Roll down
20 X<>Y
21 10
22 ÷
23 IP
24 STO "NUM"    :divide the number by 10 and store in the working register
25 X<>Y
26 STO IND "A"  :store the digits in R7 through R1, for the 7th through 1st
digits
27 STO+ 21      :sum the digits in R21
28 11           :steps 28-31 use the digit itself to point to the registers
(11
29 +            : through 17)  that will count the number of times each digit
30 1            : occurs.   For example, if the digit is a 6, a one will be
added
31 STO+ IND ST Y: to R17
32 10           :steps 32-35 recall the running sum of the digits and if
33 RCL 21       : greater than 10, branch out of the loop to get the next
34 X>Y?         : 10 digit number to check
35 GTO 03
36 DSE "A"      :decrement the counter and go get the next digit of the
current
37 GTO 01       : number
38 10           :all digits checked, the sum must be less than or equal to 10,
39 X!=Y?        : if not equal to 10, go get the next 10 digit number.
40 GTO 03
41 7            :initialize counter to loop through the routine that checks
the
42 STO "I"      : digits against the counts of the digits
43>LBL 02       :routine that checks the digits against the counts of the
digits
44 RCL "I"
45 10
46 +            :add 10 to the index
47 RCL IND ST X :recall the count of that digit
48 RCL IND "I"  :recall the digit
49 X!=Y?
50 GTO 03       :if they’re not equal, go get the next 10 digit number
51 DSE "I"      :if they are equal, decrement the counter to check the next
digit
52 GTO 02       : vs. the count.
53 PRON         :If all digits are equal, i.e. the checking routine is not
branched
54 PRV "LB"     : out of, this must be the answer!  Print it out and beep to
55 PROFF        : announce completion
56 BEEP
57 STOP
58>LBL 03       :routine to increment the 10 digit number to get the next one
to
```

```
59 1E3          : check.  Since there can be no sevens, eight, or nines, the
last
60 STO+ "LB"    : three digits are zero and the increment is 1000.
61 RCL "LB"
62 RCL "UB"     :check the new 10 digit number against an upper bound to
63 X!=Y?        : insure that the program stops someday.
64 GTO "FIND2"  :go to the beginning if the upper bound has not been reached
65 PRON
66 PRX          :if the upper bound is reached, print it out and stop to
indicate
67 PROFF        : that no solution was found.
68 END
```

1 – due credit thankfully given to the late, great Douglas Adams.

Re: S&SMC#10: My Original Solutions & Comments

*Posted by **Arnaud Amiel** on **8 June 2005, 1:52 p.m.**, in response to Re: S&SMC#10: My Original Solutions & Comments, posted by Jeff O. on 8 June 2005, 1:15 p.m.*

I was using a recursive algorithm and was also worried that it might not converge to a solution. It happened that 0000000000 did converge and all the few numbers I tried so I assumed it would always converge and did not check for non convergence. I was actually thinking of writing a little C program to go through all 10 digit numbers and see which did converge and which did not. But I might not do it before next week end.

Arnaud

Re: S&SMC#10: My Original Solutions & Comments

Posted by ***Valentin Albillo*** *on* ***9 June 2005, 5:14 a.m.****, in response to Re: S&SMC#10: My Original Solutions & Comments, posted by Jeff O. on 8 June 2005, 1:15 p.m.*

Hi, Jeff:

Jeff posted:

"First, I'll second (or third, or fourth..) the thanks to you for the S&SM Challenges."

Why, thank you, you're all so very kind. I certainly appreciate it when you people post your inputs to my S&SMCs, because that's the only way I have to know that my humble efforts are indeed reaching you and contributing to the share of enjoyment and knowledge this MoHP Forum actually is.

"No clever tricks using matrix capabilities, no knowledge that a recursive approach would lead to a solution. Just brute force"

Actually, the iterative method I used isn't a trick, but a genuine, useful method to solve a large variety of equations than can be expressed as **f(x) = x**, specially when said equations aren't amenable to other faster, well-known methods such as Newton's method, etc.

The only 'bright' idea needed was to recognize that solving this puzzle was tantamount to solving f(N)=N, where f(N) was the function counting the digits in a 10-digit N and returning them assembled as another 10-digit number.

As for brute force, this would be a straightforward brute force approach for the HP-71B, using a minimum of heuristics, namely that the digits of the number must add exactly to 10:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),M(9)
20 FOR A=9 TO 1 STEP -1 @ N(0)=A
30 FOR B=9 TO 0 STEP -1 @ N(1)=B @ S=A+B @ IF S>10 THEN 210
40 FOR C=5 TO 0 STEP -1 @ N(2)=C @ T=S+C @ IF T>10 THEN 200
50 FOR D=3 TO 0 STEP -1 @ N(3)=D @ U=T+D @ IF U>10 THEN 190
60 FOR E=2 TO 0 STEP -1 @ N(4)=E @ V=U+E @ IF V>10 THEN 180
70 FOR F=2 TO 0 STEP -1 @ N(5)=F @ W=V+F @ IF W>10 THEN 170
80 FOR G=1 TO 0 STEP -1 @ N(6)=G @ X=W+G @ IF X>10 THEN 160
90 FOR H=1 TO 0 STEP -1 @ N(7)=H @ Y=X+H @ IF Y>10 THEN 150
100 FOR I=1 TO 0 STEP -1 @ N(8)=I @ Z=Y+I @ IF Z>10 THEN 140
110 N(9)=10-Z @ MAT M=ZER @ FOR J=0 TO 9 @ M(N(J))=M(N(J))+1 @ NEXT J
120 MAT M=M-N @ IF RNORM(M) THEN 140
130 FOR J=0 TO 9 @ DISP N(J); @ NEXT J @ END
140 NEXT I
150 NEXT H
160 NEXT G
170 NEXT F
180 NEXT E
190 NEXT D
200 NEXT C
210 NEXT B @ NEXT A @ DISP "No solution"
```

Running it on a physical HP-71B returns **6210001000** in 8.03 seconds (0.91 seconds under Emu71). You can very easily adapt this to your HP42S, and it will run pretty fast there.

"With time, I think I can probably figure out the workings of the 15C program. I may attempt to use matrix processing capabilities of the 42S"

It's actually quite easy to understand, and translates most easily to HP42S RPN and matrix capabilities. Try it !

"Is the procedure that you implemented mathematically guaranteed to lead to a solution?"

Yes, it is, but the actual, rigurous proof of it is quite cumbersome and lengthy. An informal argument would be as follows: the statistical distribution of the randomly generated initial guesses entirely depends on the properties of the HP-15C's RNG, and this one passes the spectral test and has a cycle length large enought that eventually every 10-digit number would be generated with probability 1. This means that, in the very worst case that only stumbling directly upon the solution would result in a 1-cycle, you're guaranteed that this would happen. Fortunately, it seems likely (and a little testing quickly demonstrates) that there are indeed many other initial guesses which quickly lead to the solution, so the probabilities increase to the point where finding one of them by chance has a large probability.

I did some empirical tests, trying out up to 10,000 random initial guesses, and as stated in a previous post, you'd only generate 13.5 numbers (average) before finding the solution, very frequently after just one or two restarts (new random initial guess). Many times I got by with only 3 numbers generated, one or two times it would need 100. And, of course, checking at most 100 numbers (out of the 9E9 possible) is incredibly fast on any decent machine, let alone checking just 13 or 3.

Thanks again for your interest and

Best regards from V.

Re: Re: S&SMC#10: Apologies hp17bII+ solution

*Posted by **Chris dean** on **13 June 2005, 4:44 p.m.**, in response to Re: S&SMC#10: hp17bII+ solution, posted by Chris Dean on 13 June 2005, 2:57 p.m.*

Sorry it should have read .....

Apologies for the lateness of this solution but after much counting of brackets I submit a program for an HP17bII+.

```
SSMC=0*L(A:9000000000)+0*SIGMA(I:0:9:1:
   0*L(B:0) + 0*L(C:ALOG(9-I)
      +0*L(D:MOD(IP(G(A)/G(C)):10) +
         +0*SIGMA(J:0:9:1:L(X:MOD(IP(G(A)/ALOG(J)):10))
            +0*IF(G(X)=I:L(B:G(B)+1):G(B)))
         +0*IF(G(D)<>G(B):
      L(A:G(A)-G(D)*G(C)+G(B)*G(C)):G(A)))
   + G(A)
```

Do not forget to register the variables i.e. Y=A+B+C+D+X. The result is obtained after 5 interations. This shows what can be done on a HP17bII+ with a reasonable amount of patience!

Chris Dean 12345

Re: Re: S&SMC#10: Apologies hp17bII+ solution

*Posted by **Valentin Albillo** on **14 June 2005, 9:27 a.m.**, in response to Re: Re: S&SMC#10: Apologies hp17bII+ solution, posted by Chris dean on 13 June 2005, 4:44 p.m.*

Hi, Chris:

Chris posted:

"This shows what can be done on a HP17bII+ with a reasonable amount of patience!"

Indeed ! Truly amazing effort on your part, I was expecting entries in RPN, RPL, BASIC, Assembler, ... but an entry for a 17BII+ was most unexpected.

Give the man a cigar ! :-) Thanks for your interest, and

Best regards from V.

Re: S&SMC#10: hp17bII+ solution

Posted by ***Chris Dean*** *on* ***13 June 2005, 2:57 p.m.***

Apologies for the lateness of this solution but after much counting of brackets I submit a program for an HP17bII+.

```
SSMC=0*SIGMA(I:0:9:1:
   0*L(B:0) + 0*L(C:ALOG(9-I) + 0*L(A:9000000000)
      +0*L(D:MOD(IP(G(A)/G(C)):10) +
         +0*SIGMA(J:0:9:1:L(X:MOD(IP(G(A)/ALOG(J)):10))
            +0*IF(G(X)=I:L(B:G(B)+1):G(B)))
         +0*IF(G(D)<>G(B):
      L(A:G(A)-G(D)*G(C)+G(B)*G(C)):G(A)))
   + G(A)
```

Do not forget to register the variables i.e. Y=A+B+C+D+X. The result is obtained after 5 interations. This shows what can be done on a HP17bII+ with a reasonable amount of patience!

Chris Dean 12345

Re: S&SMC#10: My Original Solutions & Comments

Posted by ***Jeff O.*** *on* ***13 June 2005, 7:36 a.m.****, in response to Re: S&SMC#10: My Original Solutions & Comments, posted by Valentin Albillo on 9 June 2005, 5:14 a.m.*

> Quote:
>
> ---
>
> Running it on a physical HP-71B returns **6210001000** in 8.03 seconds...
>
> ---

Valentin,
I typed your program into my 71B. After about 10 seconds of running with no response, I figured I must have made an error somewhere. However, I let it continue running. Some time later, I looked at the display. It had stopped running and was displaying the answer. I ran it again and timed it, finding that it takes approximately 1 minute, 45 seconds to find the answer. Is there some reason your physical 71B would run the program 12 times faster than mine?