HP Forum Archive 16

[Return to Index | Top of Index]

Short & Sweet Math Challenge #17: The Feast !

*Message #1 Posted by **Valentin Albillo** on 13 Sept 2006, 1:10 p.m.*

Hi all,

Several months have merrily passed by since I posted my previous S&SMC#16 and now it's as good time as any to post my brand-new **Short & Sweet Math Challenge #17: The Feast!**, arranged meal-like, so to say. Dust off once more what's left of your HP-calc programming skills (if any) and enjoy ... Bon appetit !

Important: Rules & Notes:

- In all cases, you must write a program for your chosen HP handheld preferably, but other brands are also acceptable as long as they're handheld (but no PDAs), emulators/simulators for such models are also welcome. Any programming language is acceptable as long as it actually runs in your chosen handheld.
- Please **absolutely refrain** from posting just the solutions, **actual code written by yourself *is* mandatory** . Googling solutions out and copy-posting them is pretty *lame* and only serves to spoil the challenge for others and to make blatantly public your utter lack of programming skills and/or sheer disrespect for fair rules and this forum's visitors.

Appetizer: More power to you

"Write a program to find the first exact power of 2 (up to 2^{2500}, say) that begins with a given sequence of up to four digits"

For instance, if the given sequence is **102** your program should output **10**, as $2^{10} = \underline{\mathbf{102}}4$. You can either give just the required exponent (**10**), or also output the power itself (**1024**), for extra *élan*.

Use your program to find (and output) the first exact power of 2 which begins with **1000, 1234, 1313, 2006, and 1776**.

I'll give both a fast *3-line* HP-71B program for the easier case, and a still simple, fairly fast program for the more elaborate second case.

Main course: Le petit homage to the 41 !

"Find a 41-digit number consisting solely of the digits 4,1 which can be halved at least 41 times. The solution is *unique !*"

Important note:

The description above and below has been duly amended to correct the original wording which erroneously said *"precisely 41 times"* where actually it should read *"at least 41 times"*. This was first detected by GE and Marcus von Cube. Thanks a lot for pointing it out and sorry to everyone for any inconveniences.

Of course, *"halved N times"* means that the number can be halved at least N *consecutive* times. For instance, 100 can be halved 2 times: (1) 100/2 = 50, (2) 50/2 = 25, as 25 can't be halved any further to yield an integer result.

Despite its looks it's actually fairly easy, and to prove it I'll give a very short, *7-line* HP-71B program which finds out and outputs the only solution in less than 4 minutes in a physical HP-71B (and less than a second under Emu71!!).

Le Dessert: Pi & other delights

A recent HP-15C mini-challenge o'mine featured an infinite, convergent series for Pi/4, namely:

```
Pi/4 = 1 -1/3 +1/5 -1/7 + 1/9 - 1/11 + 1/13 - ...
```

Now this infinite series though convergent, if slowly, is a typical case of what's known as an *alternating series*, because its terms steadily alternate between positive and negative. Furthermore, infinite alternating series like this one, even if convergent, are only *conditionally convergent*, which means that their value depends on the exact order in which its terms are summed up.

So by varying their order, and even though you're still adding up each and every term exactly once and only once, lo and behold, you can get the series to add up to *any arbitrary real value* at all, including Pi/4, zero, negative values, or even make it diverge to infinity !

This understood, **"you must write a program which, given an arbitrary value, will output the exact order in which the terms of this particular series have to be summed up so that the series sums to precisely that given value".**

As all the terms have "1" as numerator, your program needs to output just the *denominators* to fully identify each term, in the required order. For instance, assuming you've just written a version for the HP-15C, it should work like this example (User mode), in which the sum must be precisely Pi/4:

```
Pi/4, [A] ->  1  -> -3  ->  5  ->  -7 ->  9  ->  -11  -> ...
```

where the program can continue indefinitely, if desired, to produce as many terms as the user cares for. In this example, the output are the denominators, so the order is:

```
Pi/4 =  1/1 -1/3 + 1/5 -1/7 +1/9 -1/11 ...
```

as expected. Use your program with the following cases, outputting a few dozen terms or so:

```
Sum = Pi/4, Pi/5, e/4, Sin(1), Cos(1), 1, 0, 1-1/3+1/5, .71, .41
```

I'll give both my original solutions, namely a *3-line* program for the HP-71B and a *30-step* version of it for the HP-15C, both taking negligible time.

That's all. I'll post my original solutions next week, so you've got plenty of time to work yours out. Now respect the rules, sharpen your ingenuity, post your achievements, and ... Enjoy !

Best regards from V.

Edited: 18 Sept 2006, 8:52 a.m. after one or more responses were posted

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #2 Posted by **Bruce Horrocks** on 13 Sept 2006, 2:18 p.m.,*
in response to message #1 by Valentin Albillo

Quote:

Several months have merrily passed by since I posted my previous S&SMC#16 and now it's as good time as any to post my brand-new **Short & Sweet Math Challenge #17: The Feast!**

It's not such a good time - we're all off to HHC2006. You won't hear anything here except the wind whistling through the tumbleweed!!

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #3 Posted by **Valentin Albillo** on 13 Sept 2006, 2:53 p.m.,*
in response to message #2 by Bruce Horrocks

Hi, Bruce:

Bruce posted:

"It's not such a good time - we're all off to HHC2006."

Gosh, you're right ! I hadn't noticed !! :-(:-(

I'll delete my post *immediately* so please hold your breath while I'm at it, it'll take just a moment ...

Best regards from V.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #4 Posted by **Namir** on 13 Sept 2006, 3:39 p.m.,*
in response to message #2 by Bruce Horrocks

Ditto Bruce .... the only pi HHC2006 attendee will have are ones served by San Jose cantinas (aka restaurants).

Namir

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #5 Posted by **Paul Dale** on 14 Sept 2006, 3:19 a.m.,*
in response to message #1 by Valentin Albillo

I've got an inelegant solution to the first part on the HP49g+:

```
<< 512 DO 2 * DUP ->STR 1 4 SUB OBJ-> UNTIL 3 PICK == END >>
```

The output is the power itself rather than the exponent. Runtime for the smallest case beginning 2006... is under a minute. The largest case beginning 1000... takes about three minutes to solve.

I'm guessing that this is faster than attacking the problem mathematically :-)

Converting to get the exponent as asked for is left as an exercise for the reader :-) Either figure it out from the number (beware the limited real exponenet range) or add a counter to the program.

- Pauli

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #6 Posted by **Bram** on 14 Sept 2006, 6:49 a.m.,*

in response to message #1 by Valentin Albillo

So you cooked some bits for a nice bite, Valentin.

Well, as I'm experienced with powers of 2 (remember? ;-) I made this program as a starter.

```
LBL V
STO S   ; sequence of digits
LOG
IP
STO D   ; number of digits of sequence
0
STO P   ; power
1
STO N   ; number, 2^P
LBL A
1
STO+ P
2
STO* N
RCL N
ENTER
LOG
RCL D
-
IP
10^x
/
IP
RCL S
x#y?
GTO A
RCL P
STOP
```

Not really brilliant, but it will do, although you'll need something like an HP-32SII because of the very large numbers. Yet only 3 tasks can be computed, the other 2 give an overflow.

1234 XEQ V yields 1545 so 2^1545 starts with 1234
2^1183 starts with 1313
2^682 starts with 2006 (takes 1.5 minutes)
and even 102 XEQ V yields 10 as required

I'll have to run this program in Free42 to get the other two answers. (or find another algorithm)

Now, let's digest the material for a while before continuing ...

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #7 Posted by **GE** on 14 Sept 2006, 11:59 a.m.,*
in response to message #6 by Bram

My first try at a SSMC, while all were interesting and sometimes mind-bending !!

Challenge 1 :
Apologies for putting a program for a Casio calculator here, that's all I have around today. Here goes :

```
?->A:log A->B:0->C
Lbl I:C+1->C:Clog 2->D
A=/=Int 10^(B+Frac D)=>Goto I
C
```

44 bytes plus 32 bytes overhead, and =/= above is the sign 'different from'.
It uses the log(2^N) instead of 2^N to avoid overflows. This gives 1000=>2136, 1234=>1545, 1313=>1183, 2006=>682, 1776=>2087 in under 30 seconds in all cases.
The theory is to calculate the first few digits of 2^N using the floating point approximation, which should be precise enough for this (not proven here). I wonder if ALL integer values can be found as the beginning sequence of a power of two, my guts feeling being a big NO, but...

Challenge 2.
Only ideas here. Only the leftmost N digits are relevant for the first N halvings, so at each step based on the parity of the current value of X/2^N we can determine the next digit as being 4 or 1, there is one solution and there is only one, we can build it digit by digit.
Note that the two digits must have different parity but could be anything. Note also that a N digits number cannot be halved more than log(N)/log(2) times, so there is also a solution for halving that 41-digit number (less than 10^42) made of two different parity digits exactly P times for P between zero and 139 times (because 42/log2 = 139,52).
Sorry I had not the juice to build the multi-precision program which would spit each digit, maybe later.

Challenge 3.
Ideas only, I'll check before posting here.

And again many thanks to Valentin for these really interesting, fun puzzles.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #8 Posted by **Kiyoshi Akima** on 14 Sept 2006, 7:09 p.m.,*
in response to message #1 by Valentin Albillo

Brute force and ignorance on part one (exponent only).

```
102  => 2^10
1000 => 2^2136
1234 => 2^1545
1313 => 2^1183
2006 => 2^683
1776 => 2^2087
```

And some boundary cases (not absolutely sure about the last two, but they're what these programs give) (I know these go beyond the specs):

```
1    => 2^0
9999 => 2^13301
9975 => 2^9999
```

1542 is a somewhat interesting case.

For the HP25 (minor and obvious changes for almost any RPN machine(?)):

```
01 STO 1
02 LOG
03 INT
04 STO 2
05 1
06 CHS
07 STO 3
08 2
09 LOG
10 STO 4
11 1
12 STO + 3
13 RCL 3
14 RCL 4
15 *
16 ENTER
17 FRAC
```

```
18 x<>y
19 INT
20 RCL 2
21 x<y?
22 x<>y
23 Rdown
24 +
25 10^x
26 EEX
27 CHS
28 4
29 +
30 INT
31 RCL 1
32 x#y?
33 GTO 11
34 RCL 3
```

HP-48GX (and most RPL machines(?)):

```
<<
 DUP LOG IP -1 2 LOG -> n g i t
 <<
  DO
   UNTIL t 'i' INCR * DUP FP SWAP IP g MIN
   + ALOG 6 RND IP n ==
  END
  i
 >>
>>
```

HP-71B:

```
10 INPUT N @ L=IP(LOG10(N)) @ I=-1 @ T=LOG10(2)
20 I=I+1 @ P=I*T @ Q=FP(P)+MIN(IP(P),L) @ P=IP(10^Q+.0001) @ IF P#N THEN 20 ELSE PRINT I
```

Edited: 14 Sept 2006, 7:11 p.m.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #9 Posted by **GE** on 15 Sept 2006, 4:52 a.m.,*
in response to message #8 by Kiyoshi Akima

Hello, some more news :

Challenge 1 :

Dug out an HP28S and typed the following program which gives the same results. It is the simplest I can do, achieved as usual by avoiding stack monkey business and the ugly STO syntax. This program also shows how far we've gone since HP28 days, as that machine is very significantly slower than the Casio... A wonderful calculator IMHO despite RPL. Note that internal precision allows for no rounding and still correct results !
<< DUP LOG IP -> A B << 0 DO 1 + UNTIL
DUP 2 LOG * FP B + ALOG IP A == END >> >>

Challenge 2 :

As stated a direct approach is possible and gives the following program on the Casio. This version is in single precision, and thus works only up to 9 digits.
8->B:{4,1}->List 1
0->N:0->M
For 0->D To B-1
List 1[1+2Frac (N+2)]->A
(N+A*5^D)/2->N
M+A*10^D->M:Next
List 1[2-2Frac (N+2)]->A
M+A*10^(D+1)->M:M

I extended it in a 'brute force' way to multi precision and that gave an horrible 600 bytes program that I will NOT write down here, this program finds the solution but per the rules I will not show the solution only (it begins with 111111...).
Just for fun here is the 67-digit number containing only digits 6 and 7 which is exactly 67 times halvable :
7676776776777667766677676667677677767676677776667667776667667776, found in about 10 seconds on the Casio 85 (the currently fastest Casio). You just have to change the parameters in the first line, here as :
67->B:{6,7}->List 1

Challenge 3 :

Same idea as Dale, congratulations.
Note that a divergent series remains divergent if you remove a finite number of terms, this suffice to validate the procedure.

Waiting for SSMC 18 !!

Magnificent, but ...

*Message #10 Posted by **Valentin Albillo** on 15 Sept 2006, 5:55 a.m.,*
in response to message #9 by GE

Hi, GE:

GE boldly posted:

"Just for fun here is the 67-digit number containing only digits 6 and 7 which is exactly 67 times halvable : 7676776776777667766677676667677677676767667777666766777666766667776, found in about 10 seconds on the Casio 85"

> Very funny, indeed, but either your Casio isn't working as it should, or your programming is slightly faulty, or you missed school the day counting was taught ! :-) :-) Because your *"67-digit number"*, as posted by you, is actually a **68**-digit number, and further, it isn't 67 times halvable but only **15**.
>
> I can't comment on your 41-digit result or your multi-precision program capable of computing it as you provided neither. Perhaps a little rechecking is in order ? :-) All in good humor, of course, no offence whatsoever intended :-) :-).

"Waiting for SSMC 18 !!"

> Seems to me that you've still some work to do with the "main course" before asking for S&SMC #18. Unfinished business :-)
>
> Thanks for your interest and extremely kind comments, much appreciated.

Best regards from V.

Edited: 15 Sept 2006, 6:10 a.m.

Re: Magnificent, but ...

*Message #11 Posted by **GE** on 15 Sept 2006, 12:02 p.m.,*
in response to message #10 by Valentin Albillo

Aaargh, you are right !!!!!! Time to search for my PI button...
"I'll be back", showing the program after some much-needed debugging.
And no offence taken, that doesn't fit here.

Re: Magnificent, but ...

*Message #12 Posted by **GE** on 18 Sept 2006, 4:38 a.m.,*
in response to message #11 by GE

Hello again.
I killed the 3 bugs in my program but (as you noted) what I have found is a way to very quickly find N+1-digits numbers which are N-halvable (and containing only two digits which are different modulo 2).
This can be modified easily to give N-digit numbers which are *AT LEAST* N-halvable, but this method doesn't guarantee it is not N+1-halvable (or more).
And in the case of 41, alas, the 41-digit number I get is 42-halvable, so (per the rules) I doesn't solve the problem and I can publish it :
44111111411444111411141441444411441414144.
Just for fun here is a correct 65-digit, 65-halvable number made of only the digits 6 and 5 (calculation time about 15 seconds) :
66656565566565655566565665666666566656655655556556566656666566656
This also works for the "HP-numbers" 12, 21, 25, 32, 34, 45, 49, 81 and 85.
Since you said that there is a solution for the 41 case, I have not tried to prove that this number doesn't exist... enough made a fool of myself... and writing a brute force program (testing the 2^41 possibilities) doesn't taste good. Out of theoretical ideas for now, I leave it to others to find the shortcut.

Ouch! You proved me wrong!

*Message #13 Posted by **Marcus von Cube, Germany** on 18 Sept 2006, 5:11 a.m.,*
in response to message #12 by GE

Hi GE,

I should have checked my own solution better! The number my program gave is exactly the one you posted here and it is 42 time halvable.

:-(

Back to the drawing board!

Marcus

My fault.

*Message #14 Posted by **Valentin Albillo** on 18 Sept 2006, 6:13 a.m.,*
in response to message #13 by Marcus von Cube, Germany

Hi, Marcus & GE:

Marcus posted:

"I should have checked my own solution better! The number my program gave is exactly the one you posted here and it is 42 time halvable."

> No, the error's actually mine, I said *"precisely 41 times"* where I should have stated *"at least 41 times"*, because as both of you correctly pointed out, the solution number is indeed divisible by 2 more than 41 times.
>
> Matter of fact, there would be no solution if the number had to be divisible by 2 exactly 41 times, as there's but a single choice for the digit, 4 or 1, at each step. If putting a 4 makes the number divisible 42 times, putting a 1 instead would make it divisible just 40 times, and there are no other possibilities.
>
> Sorry for the inexact wording, I'll correct my original posting when the thread's finished for good, so that it doesn't get archived with the error in place, with an explanation for the need of the correction and full credit to you.

Thanks and best regards from V.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #15 Posted by **Paul Dale** on 16 Sept 2006, 6:17 a.m.,*
in response to message #9 by GE

> Quote:
>
> Challenge 3 :
> ---
> Same idea as Dale, congratulations.

Pauli or Paul please :-)

Valentin's HP-15C program is some thirty steps which is quite a bit longer than my HP-12C progam (twenty one) or a naive conversion of this to the HP-15C (twenty four). Even skipping the hard coded initial '1' term doesn't bloat the program enough. I wouldn't be at all surprised if Valentin is attacking this problem differently.

- Pauli

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #16 Posted by **Valentin Albillo** on 16 Sept 2006, 11:19 a.m.,*
in response to message #15 by Paul Dale

Hi, Paul:

Paul posted:

"Valentin's HP-15C program is some thirty steps which is quite a bit longer than my HP-12C progam (twenty one) [...] I wouldn't be at all surprised if Valentin is attacking this problem differently."

> No, that's not it. It's simply that my HP-15C program is a rough translation of my original 3-line HP-71B BASIC program, without further thoughts or optimizations, and the HP-71B solution itself is also pretty straightforward, the first thing that came to mind. Neither are brilliant pieces of code, after all it's just the "dessert" ! :-)

Thanks for your interest and

Best regards from V.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #17 Posted by **Paul Dale** on 14 Sept 2006, 8:43 p.m.,*
in response to message #1 by Valentin Albillo

The third challenge.

I'm aware of the approch mentioned in this paper (PDF file). I'm reasonably confident that something similar could be done in this case.

Since following that path would be no fun, I present a *twenty-one* step HP-12c program that achieves the goal in a different manner:

```
01  STO 0
02  1
03  STO 1
04  STO 2
05  STO 3
06  PSE             Display the first 1 term
07* RCL 0           Main loop
08  RCL 3           Compare current total
09  x<=y?             against target number
10  GTO 18
11  4               Above target
12  STO- 2          Add next negative term
13  RCL 2
14* PSE             Display the term
15  1/x             And add it into our sum
16  STO+ 3
```

```
17  GTO 07
18* 4               Below target
19  STO+ 1          Add next positive term
20  RCL 1
21  GTO 14
```

The basic premesis is that if the running sum is greater than the target, we add a negative term and if it is less, we add a positive one. I think a proof of convergence should be fairly straightforward but I'll leave that (or proof that this is wrong) as an exercise...

I always start with a positive 1 term because that made the code a little shorter.

- Pauli

Edited: 14 Sept 2006, 8:46 p.m.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #18 Posted by **Fernando del Rey** on 15 Sept 2006, 9:58 a.m.,*
in response to message #1 by Valentin Albillo

This is just another RPN version to the first challenge, but I think it may be faster than others posted earlier.

I have tried it on the 42S, but it's very easily adaptable to other models.

```
01 LBL AB
02 1
03 STO 00
04 STO -00
05 LBL 01
06 1
07 STO +00
08 Rdown
09 ENTER
10 +
11 IP
12 X=Y?
13 GTO 00
14 Rdown
15 LASTX
16 X<Y?
17 GTO 01
18 0.1
19 *
20 GTO 01
```

```
21 LBL 00
22 RCL 00
23 END
```

Now it's time to try to tackle the other two challenges.

Thanks, Valentin, for keeping us entertained over the weekend !

You're welcome, thanks for participating ! :-) [NT]

*Message #19 Posted by **Valentin Albillo** on 15 Sept 2006, 10:15 a.m.,*
in response to message #18 by Fernando del Rey

Best regards from V.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #20 Posted by **Marcus von Cube, Germany** on 15 Sept 2006, 1:06 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

Here is an RPL solution to the main course: Le petit homage to the 41! I didn't read any of the other postings before in order not to get spoiled by someone elses genius ;-)

I did it on my new 50g in exact mode because it can easily cope with 41 digit integers.

But first a little theory how it works:

If you want to decide whether a number is divisible by 2, just check the last digit: If it is divisible by 2, the whole number is.

If you want to decide whether a number is divisible by 4, just check the last two digits: If they (as a number) are divisible by 4, the whole number is.

More general: If you want to decide whether a number is divisible by 2^n, just check the last n digits: If they (as a number) are divisible by 2^n, tho whole number is.

Now I start building the number from back to front. Since the digits can only be 1 or 4, the goal is to prepend one of these to the result found so far and check for divisibility by the proper power of 2. The result is unique because exactly one of the digits one or four yields to a number which is evenly divisible (the difference between the two possible results is $3*10^{n-1}$ which is *not* divisible by 2^n.)

To ease the algorithm, the program starts with n=2 and result=4. It takes the number of places (and hence the power of 2) from the stack. Minimum input is 2, smaller numbers are treated as 2.

Here is the code:

```
%%HP: T(3)A(R)F(.);
\<<
  1 2 4 \-> n f d r
  \<<
    2 n START
      10 'f' STO*
      2 'd' STO*
      f r +      @ First try with digit 1
      DUP d MOD
      IF 0 \=/ THEN
        f 3 * +  @ Digit must be 4
      END
      'r' STO
    NEXT
    r            @ result goes to stack
  \>>
\>>
```

The result for 41 is:

```
4411111141144411141111414414444411441414144
```

The computing time is within seconds.

[Edited: The given result is not *exactly* but *at least* n times halvable. Valentin is about to correct his original posting to match this request.]

Marcus

Edited: 18 Sept 2006, 4:34 p.m. after one or more responses were posted

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #21 Posted by **GE** on 18 Sept 2006, 5:06 a.m.,*
in response to message #20 by Marcus von Cube, Germany

Sorry Marcus, this number is 42-halvable...

I am currently trying to understand why a 41-halvable number could possibly **exist** at all, so I'm waiting for :
- Valentin to publish his number OR
- my brain to start working
whichever comes first... and you can guess which will come first !!
By the way, this SSMC is great.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #22 Posted by **Marcus von Cube, Germany** on 18 Sept 2006, 5:13 a.m.,*
in response to message #21 by GE

GE,

our postings crossed eachother, I've just found it out! I have to go back to my ideas.

Marcus

Please read message #14 above. [NT]

*Message #23 Posted by **Valentin Albillo** on 18 Sept 2006, 6:22 a.m.,*
in response to message #21 by GE

Best regards from V.

Re: Please read message #14 above. [NT]

*Message #24 Posted by **Marcus von Cube, Germany** on 18 Sept 2006, 7:32 a.m.,*
in response to message #23 by Valentin Albillo

Valentin,

I've edited my posting again to refer to your correction.

Marcus

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #25 Posted by **Marcus von Cube, Germany** on 18 Sept 2006, 5:33 a.m.,*
in response to message #20 by Marcus von Cube, Germany

Replying to myself... (thinking aloud)

The problem with my "solution" is that the last digit is a 4 and hence divisible by 2^2. Now following the algorithm, I always get numbers which are divisible by another power of two. The result is that all numbers my program produces are divisible by 2^{n+1} where n is the number of digits.

Does this mean, there is no solution or is my algorithm wrong?

Marcus

Please read message #14 above. [NT]

*Message #26 Posted by **Valentin Albillo** on 18 Sept 2006, 6:21 a.m.,*
in response to message #25 by Marcus von Cube, Germany

Best regards from V.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #27 Posted by **GE** on 18 Sept 2006, 11:04 a.m.,*
in response to message #25 by Marcus von Cube, Germany

> The problem with my "solution" is that the last digit is a 4
> and hence divisible by 22.

Yuckkkk !!! NOT ALL (hum) numbers ending in 4 are divisible by 4.
Actually, EVERY N-digit number divisible by 2^N containing only digits 4 and 1 and whose first digit is 1 is divisible by NO MORE than 2^N.
Those beginning with 4 are divisible by 2^(N+1).

Hopefully my mother has no Internet access.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #28 Posted by **Valentin Albillo** on 18 Sept 2006, 11:41 a.m.,*
in response to message #27 by GE

Hi, GE:

GE boldly posted:

"Actually, EVERY N-digit number divisible by 2^N containing only digits 4 and 1 and whose first digit is 1 is divisible by NO MORE than 2^N."

Nope. For instance, 144 is a 3-digit number divisible by 2^3 (8), containing only digits 4 and 1, and whose first digit is 1.

Yet it *is* divisible by 2^(3+1), as 144 = 16 * 9.

"Hopefully my mother has no Internet access."

Let's hope ... ;-)

Best regards from V.

Problem 3

*Message #29 Posted by **Crawl** on 15 Sept 2006, 2:56 p.m.,*
in response to message #1 by Valentin Albillo

Problem 3 seems to be the easiest.

The principle here is just to add a term if the sum is less than the desired value, subtract if it's greater, and go to the next term (whichever the case may be) if they're equal.

You put the desired value on the stack, and select "okay" from the choose boxes as long as you want more terms. Selecting "cancel" will end the program, with the desired value on level 2, and the approximation using the sum, for comparison, on level 1.

```
%%HP: T(3)A(R)F(.);
\<< 0. 0. 0. \-> V P N S
  \<< 1.
    DO DROP V S >
      IF
      THEN P 4. * 1. + DUP INV S + 'S' STO P 1. + 'P' STO
      ELSE V S <
        IF
        THEN N 4. * 3. + NEG DUP INV S + 'S' STO N 1. + 'N' STO
        ELSE V S ==
          IF
          THEN P N \<=
            IF
            THEN P 4. * 1. + DUP INV S + 'S' STO P 1. + 'P' STO
            ELSE N 4. * 3. + NEG DUP INV S + 'S' STO N 1. + 'N' STO
```

```
            END
          END
        END
      END \->STR DUP "{{" SWAP "}}" + + OBJ\-> 1. CHOOSE
    UNTIL NOT
    END V S
  \>>
\>>
```

The first few terms in the sequences for various desired values:

```
pi/4

1, -3, 5, -7, 9, -11, 13, -15, 17, -19, 21, -23

pi/5

1, -3, -7, 5, -11, -15, 9, -19, 13, -23, -27, 17

e/4

1, -3, 5, -7, -11, -15, 13, -19, -23, 17, -27, -31

Sin(1)

1, -3, 5, -7, 9, 13, -11,  17, -15, 21, -19, 25

Cos(1)

1, -3, -7, 5, -11, -15, -19, 9, -23, -27, -31, 13

1

1, -3, 5, 9, 13, -7, 17, 21, -11, 25, 29, -15,  33
```

```
0


1, -3, -7, -11, -15, -19, -23, -27, -31, -35, -39,
-43, -47, -51, -55, -59, -63, -67,  -71, -75, 5


1-1/3+1/5


1, -3, 5, -7, 9, 13, -11, 17, -15, 21, 25, -19


.71


1, -3, 5, -7, -11, 9, -15, 13, -19, 17, -23,  -27, 21


.41


1, -3, -7, -11, -15, 5, -19, -23, -27, -31, 9, -35
```

Edited: 15 Sept 2006, 2:58 p.m.

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #30 Posted by **Marcus von Cube, Germany** on 16 Sept 2006, 11:08 a.m.,*
in response to message #1 by Valentin Albillo

Here is my attempt at the appetizer.

It's another RPL program but the logic should be portable to almost any programming language. The algorithm does not test all powers of 2 against the given number x but it computes the matching exponent r of two directly (r = "result"):

$$r := \text{CEIL}(\log_2(x * 10^n))$$

It then checks, if

$$x == \text{IP}(2^r / 10^n)$$

10^n

n is a shift factor that is increased in a loop until a match is found.

In fact, all formulas are converted to logarithmic form. This avoids large numbers:

$r := CEIL(\log_2 x + n * \log_2 10)$

$x == IP(2^{r - n * \log_2 10})$

$\log_2 10$ is a precomputed constant.

On the 50g, extended precision integer arithmetic allows to display the exact power of two together with the final result.

The results are exactly the same as already posted by Kiyoshi Akima. And I must agree that 1542 is an interesting case :-)

Here is the code:

```
%%HP: T(3)A(R)F(.);
\<<
  \->NUM DUP LN 2. LN /
  10. LN 2. LN /
  0. 0.
  \-> x l2x l2t n r           @ l2x and l2t are precomputed logarithms
  \<<
    WHILE r 2500. <
    REPEAT
      l2t n *                 @ logarithm of shift factor
      DUP l2x + CEIL 'r' STO  @ compute exponent of two
      r SWAP - 2. SWAP ^ IP   @ 2nd formula from above
      IF x ==
      THEN
        r \->Q DUP 2 SWAP ^   @ format result
        1.E99 'r' STO         @ make loop terminate
      ELSE
        'n' INCR DROP         @ try next power of ten
      END
    END
  \>>
\>>
```

Running times are about 20 seconds for the greater results.

Marcus

Re: Short & Sweet Math Challenge #17: The Feast !

*Message #31 Posted by **Paul Dale** on 17 Sept 2006, 12:53 a.m.,*
in response to message #30 by Marcus von Cube, Germany

Although not for a calculator, here is the same algorithm written in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


// Compile using: gcc -o x x.c -lm -std=c99 -O


int main(int argc, char *argv[]) {
        unsigned int x = atoi(argv[1]?:"0");
        int n;


        if (x == 0)
                return 1;


        long double log2_10 = log2l(10);
        long double log2_x = log2l(x);
        long double r = 0;


        for (n=0; r<1000000; n++) {
                // long double n10 = powl(10.0, n);
                long double nlog2_10 = n * log2_10;
                r = ceill(log2_x + nlog2_10);
                long double t = truncl(powl(2, r - nlog2_10));


                if (x == t) {
                        printf("%Lg\n", r);
                        return 0;
                }
        }
        printf("failed to locate solution\n");
        return 0;
}
```

Run time is rather small :-)

Using this, I've verified that all five digit or fewer leading sequences can be found. For each length, here is a table of the last sequence to be found and the power of two where it occurs.

```
len    digit(s)   power
 1         9      2^53
 2        97      2^279
 3       982      2^3973
 4      7629      2^28721
 5     95367      2^325127
```

There might be some merit to the earlier suggestion that any integer could be found in the leading digits of a power of two.

- Pauli

S&SMC#17: My Original Solutions & Comments [LOOONG]

*Message #32 Posted by **Valentin Albillo** on 19 Sept 2006, 5:28 a.m.,*
in response to message #1 by Valentin Albillo

Hi all,

Thanks once more for your interest and for the excellent inputs and kind comments, much appreciated. I must say that it's becoming increasingly difficult to come up with a worthy challenge capable of resisting your combined efforts even for a few days. I dread the day when my S&SMCs will be completely solved within a few hours after posting ! This only goes to show the tremendous amount of programming & math ingenuity displayed by this forum's gentle contributors.

That said, these are my original solutions to all three parts of S&SMC#17, plus relevant comments:

Appetizer: More power to you

"Write a program to find the first exact power of 2 (up to 2^{2500}, say) that begins by a given sequence of up to four digits"

First of all, it's fairly simple to prove that for *any* positive integer M there exists an exact power of 2 that begins with the sequence of digits which represents the number M in the decimal notation. Which is more, this is also true not only for 2 but for *any positive integer A* other than an exact power of 10, so for example, you can find N such that 3141592654^N begins with 2718281828, say.

Once we know that the feat is indeed possible for any given M, we can proceed to implement the search, and this 3-liner (97 bytes) is my original solution for the HP-71B:

```
1 INPUT M @ X=FP(LGT(M)) @ Y=FP(LGT(M+1)) @ P=LGT(2) @ FOR I=1 TO 2500
2 N=FP(P*I) @ IF N>X AND N<Y THEN DISP I,STR$(10^N);"E";STR$(INT(P*I)) @ END
3 NEXT I
```

which will accept an integer M, then will find out and print the required exponent and the corresponding power of 2 (which begins with M) in exponential notation, using decimal logarithms throughout to keep the numbers within range. Upon running it for the given cases, we construct the following table:

```
Beginning    N       2^N
-----------------------------------------
1000         2136    1.00016289353E643
1234         1545    1.23407996301E465
1313         1183    1.31366573155E356
2006          682    2.00658260481E205
1776         2087    1.77664619863E628
```

and we also try out Kiyoshi Akima's suggestion:

```
1542         1542    1.54259995373E464
```

which is indeed rather nice, as the required exponent *equals* the given beginning sequence, namely **1542**, so that we have:

$2^{\mathbf{1542}} = \mathbf{1542}.59995373 * 10^{461}$

For the longer case, outputting the resulting power of 2 in full, it suffices to append a routine to compute the exact value of 2^N using some multiprecision algorithm. I'll simply use the one I already posted as part of my *"Spring Special Challenge" (aka "April's Fool Challenge")*, where it was used to efficiently compute 2^{65536}-3, and which is readily customizable for an arbitrary power of 2. The resulting program is thus the following 426-byte quick concoction:

```
10 DESTROY ALL @ OPTION BASE 0 @ INPUT M
15 X=FP(LGT(M)) @ Y=FP(LGT(M+1)) @ P=LGT(2)
20 FOR I=1 TO 2500 @ N=FP(P*I) @ IF N>X AND N<Y THEN 30
25 NEXT I @ DISP "Not found" @ END


30 N=I @ PRINT N @ M=9 @ DIM A(1) @ K=10^M
35 A(0)=1 @ P=0 @ L=9 @ R=2^L @ FOR J=0 TO N-L STEP L @ MAT A=(R)*A @ GOSUB 65
40 IF NOT MOD(J,27) THEN DIM A(UBND(A,1)+1)
45 NEXT J @ R=MOD(N,L) @ IF R THEN MAT A=(2^R)*A @ GOSUB 65
50 J=0 @ PRINT @ PRINT STR$(A(P));
55 FOR I=P-1 TO 0 STEP -1 @ J=J+1 @ IF J=7 THEN PRINT @ J=0
```

```
60 A$=STR$(A(I)) @ PRINT RPT$("0",M-LEN(A$));A$; @ NEXT I @ END
65 FOR I=0 TO P @ A(I+1)=A(I+1)+A(I) DIV K @ A(I)=MOD(A(I),K) @ NEXT I
70 P=P+SGN(A(P+1)) @ RETURN
```

Let's try a couple of cases:

```
>RUN
 ? 2006


   682


    2006582604045247462173839524414111582012306138161916297721207009532444822043258980603663076888118153086465060751410758099754116916726609750033498676548721637708749264193895186688104155687073790462987232870 4
```

which takes just 0.43 seconds in Emu71 (some 2 min. in an actual HP-71B). Another interesting case is Kiyoshi Akima's 1542:

```
>RUN
 ? 1542


   1542


      1542599953229460856691274627858121032263739678685240928870192025001764730618754048428214415428701471784607210419768323064231946866428773327980552312250095776931336537204240143853300529867316159793181147607928204836862301349058943430055287758143685097598602010459072433682862947380235154183548264939382957037036140229901818843554708487400165408051570936224576635977531061425053843956169640835118960116646723998075416255171821448016539727616789111588483046617162711 04
```

which takes only 1.64 seconds in Emu71 and some 8 min. in a real HP-71B.

Main course: Le petit homage to the 41 !

---

*"Find a **41**-digit number consisting solely of the digits **4,1** which can be halved at least **41** times"*.

As stated, this is actually quite easy, because you can begin with just one digit (4), which is certainly divisible by 2^1 and go on adding digits *leftwards* one at a time, chosing 1 or 4 so that the resulting number is divisible by 2^2, 2^3, etc. This is almost trivial up to 10-12 digits (depending on your chosen

calc model), for instance this would be one of the simplest HP-71B implementations, valid for up to 12 digits:

```
1 DESTROY ALL @ U=4 @ V=1 @ N=U @ P=4 @ T=10
2 FOR K=2 TO 12 @ IF MOD(N,P) THEN N=N+T*V ELSE N=N+T*U
3 DISP K;N @ P=P*2 @ T=T*10 @ NEXT K


>RUN


  2  44            (divisible by 2^2,  "14" wouldn't be)
  3  144           (divisible by 2^3, "444" wouldn't be)
  4  4144
  5  14144
  6  414144
  7  1414144
  8  41414144
  9  441414144
 10  1441414144
 11  11441414144
 12  411441414144  (divisible by 2^12)
```

and afterwards, for more than 12 digits, you only need a multiprecision modulus operator, which is rather easy to implement as can be seen in my following original 7-line solution for the HP-71B, which is *general* because it will work not only for the (4,1) case but for all other solvable cases as well, namely all pairs of digits which are of different parity (i.e.: one odd, the other even, zero not included):

```
1 DESTROY ALL @ INPUT U,V @ D=U+1 @ DIM N(D),B(D) @ N(1)=U @ L=10^11
2 FOR K=2 TO 10*U+V @ W=1 @ MAT B=N @ FOR J=1 TO K @ FOR I=D TO 1 STEP -1
3 IF NOT MOD(B(I),2) THEN 4 ELSE IF I#1 THEN B(I-1)=B(I-1)+L ELSE 5
4 B(I)=B(I) DIV 2 @ NEXT I @ NEXT J @ W=0
5 C=(K-1) DIV 11+1 @ N(C)=N(C)+(V*W+U*NOT W)*10^MOD(K-1,11) @ NEXT K
6 DIM T$[11*D] @ FOR I=1 TO D @ S$=STR$(N(I))
7 T$=RPT$("0",11-LEN(S$))&S$&T$ @ NEXT I @ DISP LTRIM$(T$,"0")
```

Running it for all 20 possible cases, you'll get the following table:

```
Digits  Number
-------------------------------------------------------------------------------------------------
 2,1    112111211111212122112
 2,3    23233333323323232223232
 2,5    52555522222552222552555552
 2,7    72772772727227777272722272
 2,9    29229292922999929922222292992
 4,1    441111114114441114111414414444411441414144
 4,3    43343333334333433443444434434333433333433344
 4,5    55444545454444554545545545454455554545444544
```

```
4,7    747444444774744774474474477747744447444444447744
4,9    4944949944994449999444494444999494994494494994944
6,1    1116116116116166111166666166166666666111616666661661616111616
6,3    6633363663333666363366363633363363363366333363363633333636366336
6,5    6665656556656565556656566566666566656655655556556566656666566656
6,7    6677767667676666776766667777767666777667767777777777777666766667776
6,9    9696696666966966696699666699999699699999999666969999996996969666969 6
8,1    8888111118111811181881181881888111818818811811881181181818888181888888118181888
8,3    8833383333388838333888383383333883833338888838333388833883888888888833383333888
8,5    5858588585858588585585585888885888555888585855885855585585558885585858558885888
8,7    8787887888788778778878777888787788887778787787878887878878778888878878888787877888
8,9    9999989999889988988998889899998888889898988989988998898999988889888899989899889898998988
```

The solution for the (4,1) case is thus this 41-digit number, obtained after 4 min. in a physical HP-71B (less than 1 second in Emu71):

44111111411444111411141441444411441414144

which indeed is the only 41-digit long number consisting solely of the digits (4,1), and which can be halved at least 41 consecutive times (actually, 42!). Also, the correct solution for the digits (6,7) is:

```
6677767667676666776766667777767666777667767777777777777666766667776
```

which is 67-digits long, consisting of just the digits (6,7), and can be halved at least 67 consecutive times. Running time is 15 min. in a real HP-71B (less than 4 seconds in Emu71).

By the way, before departing let's have a look at these nice 'cousins' of our number which I've generated for your viewing pleasure, also 41-digit long and consisting solely of the digits (4,1):

```
44144444111441144114411411444444411441441 =  12009881291601907477 * 3675676972953081511133 (both prime)
41411111444114444411114141144414144444441 =  is a prime
```

and these are two equally handsome 'cousins' I've generated for the 67-digit (6,7) number:

```
6777766776766667676667667777777667667776766667766766666767676677677

    = 17512766212863416348551 * 3870186294035196970770175503659617714891268 27 (both prime)

667777677677666777667766777666767776676767666676666676676767677677 = is a prime
```

Who knows, they might come handy to test your favorite factoring or primality test algorithms, or even your multiprecision multiplication routines at the very least. Or, if you're in a little sadistic mood and you've got some students the right age, have them multiply both factors by hand and keep betting whether the next digit in the result is going to be a 6 or a 7 ! ... :-)

Le Dessert: Pi & other delights

"Write a program which, given an arbitrary value, will output the exact order in which the terms of this particular series have to be summed up so that the series sums to precisely that given value"

This is probably the easiest of the lot, and this 4-liner (147 bytes) is my original solution for the HP-71B, which is anything but optimized (*pessimized* more like!) and will output up to 100 terms, then it will display both the given value and the sum after 100 terms:

```
1 DESTROY ALL @ INPUT E @ K=100 @ P=1 @ N=2 @ S=0 @ FOR I=1 TO K
2 T=2*P-1 @ S=S+1/T @ DISP T; @ P=P+2 @ IF S<E THEN 2
3 T=1-2*N @ S=S+1/T @ DISP T; @ N=N+2 @ IF S>E THEN 3
4 NEXT I @ DISP @ DISP E,S
```

Let's run it for a few cases:

```
>RUN
 ? PI/4


   1 -3  5 -7  9 -11  13 -15  17 -19  21 -23  25 -27  29 -31  33 -35  37 -39  41 -43  45 -47  49
   -51  53 -55  57 -59  61 -63  65 -67  69 -71  73 -75  77 -79  81 -83  85 -87  89 -91  93 -95  97 ...


  .785398163398        .784148171212


>RUN
 ? PI/5


 1 -3 -7  5 -11 -15  9 -19  13 -23 -27  17 -31 -35  21 -39 -43  25 -47 -51  29 -55 -59  33 -63
 -67  37 -71 -75  41 -79  45 -83 -87  49 -91 -95  53 -99 -103  57 -107 -111  61 -115 -119  65 ...


 .628318530718        .627954521532


>RUN
 ? EXP(1)/4


 1 -3  5 -7 -11  9 -15  13 -19 -23  17 -27 -31  21 -35  25 -39 -43  29 -47  33 -51 -55  37 -59
 41 -63 -67  45 -71  49 -75 -79  53 -83  57 -87 -91  61 -95  65 -99 -103  69 -107  73 -111 -115 ...


 .679570457115        .678047886984


>RUN
 ? 1
```

```
 1 -3  5  9  13 -7  17  21 -11  25  29 -15  33  37  41 -19  45  49 -23  53  57 61 -27  65  69
 -31  73  77 -35  81  85  89 -39  93  97 -43  101  105 -47  109  113  117 -51  121  125 -55  129 ...


  1                    .998110828956


>RUN
 ? SIN(1)


 1 -3  5 -7  9  13 -11  17 -15  21 -19  25 -23  29  33 -27  37 -31  41 -35  45 -39  49  53 -43
 57 -47  61 -51  65 -55  69  73 -59  77 -63  81 -67  85 -71  89  93 -75  97 -79 101 -83  105 ...


 .841470984808        .840058963385


>RUN
 ? COS(1)


 1 -3 -7  5 -11 -15 -19  9 -23 -27 -31  13 -35 -39  17 -43 -47 -51  21 -55 -59 -63  25 -67 -71
 29 -75 -79 -83  33 -87 -91 -95  37 -99 -103  41 -107 -111 -115  45 -119 -123 -127  49 -131 ...


 .540302305868        .539956898016


>RUN
 ? 0


 1 -3 -7 -11 -15 -19 -23 -27 -31 -35 -39 -43 -47 -51 -55 -59 -63 -67 -71 -75  5 -79 -83 -87 -91
 -95 -99 -103 -107 -111 -115 -119 -123 -127 -131 -135 -139 -143 -147 -151 -155 -159 -163  9 -167 ...


 0                   -1.03546604552E-4
```

I think you get the idea: though we're adding up *each and every* term of the series *once and only once*, we can get *any* result we want by simply *chosing their order* in the addition process, which is probably rather unexpected at first sight and nicely shows what *conditional convergence of alternating series* actually means.

As for the 30-step HP-15C version, it was a direct translation of the above 71B program and it doesn't bear being featured here as much better and shorter solutions have already been provided by you.

That's all. I hope you enjoyed it all and, again, thanks for your interest and stay tuned for S&SMC#18 coming next month.

Best regards from V.

Edited: 19 Sept 2006, 5:52 a.m.

Re: S&SMC#17: My Original Solutions & Comments [LOOONG]

*Message #33 Posted by **GE** on 20 Sept 2006, 4:44 a.m.,*
in response to message #32 by Valentin Albillo

Thank you again Valentin, this SSMC was very interesting. I can't wait to see the next, it must actually be difficult to find interesting subjects.

[Return to Index | Top of Index]

Go back to the main exhibit hall