Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #1 Posted by **Valentin Albillo** on 1 Apr 2008, 10:11 p.m.*

Hi all,

Spring visits us once more and there you are, a brand-new ***Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."*** for you to tame with the invaluable help of the trusty HP calc model of your choice, whether vintage or cutting-edge, plain-vanilla RPN, powerful graphical RPL, or the HP-71B.

This time the focus is on ***simplicity***. As usual, the challenge is subdivided into *five sub-challenges* to allow for variety and gradability, roughly ordered from *"simplest"* to *"least simple"*, where "simplicity" in this context means the particular problem is either simple to enunciate, simple to solve, or both.

Also, to make it even simpler for you (should that be possible), before the Challenge proper I'm giving you *three simple training-style samples* of what lies ahead, to warm you up. So flex your math-programming muscles, finish them up in a breeze, and then you're fully ready to go berserk ramming against the real Challenge tasks.

Notes:

- Any HP model of your choice may be used but I'll suggest a **Minimum Reasonable Model (MRM)** for each problem, which is the minimum model capable of solving said problem more or less comfortably, IMO. Obviously, using a more powerful model will make life even easier, and using a lesser model might be possible or not, but a real pain in the neck in any case.
- If you use an HP-41C, you can also use any of these: Math ROM, Advantage ROM, Card Reader ROM, Printer ROM. No other ROMs or extra routines allowed. If you use an HP-71B, you can also use any of these: Math ROM, HP-IL ROM, STRNGLEX. No other ROMs or LEX files allowed.
- Using a PC or any device other than a calculator is strictly forbidden and will only result in showing off your incompetence. You can write your programs in any language supported in any HP calc and your program must run in that calc (e.g., RPN, PRL, 71B-BASIC, 71B-FORTH, GCC for RPL models, etc). Emulators are also welcome. Googling for the solutions is as lame as can be, you're undeserving of entering any challenge if you do and bring nothing but shame to this forum.

The three-pronged pre-Challenge training:

1: Simplest

Given an arbitrary 4x4 real matrix, write a program to compute and output the sum of its four eigenvalues, real and/or complex.

For instance, given the 4x4 matrix:

```
 0.983142345  0.283197300 -4.115518475 -1.660698552
-0.421961836  1.757520105 -3.012345502 -1.674587499
-3.970397996 -1.332130133  3.166741493 -0.294702519
 4.881668095 -3.221150835  2.73305616  -0.108431778
```

your program must compute and output the value **5.798972165**, because its four eigenvalues happen to be:

$e_1 = 2.00989376355$
$e_2 = -1.55106880361 - 2.09930635319\ i$
$e_3 = -1.55106880361 + 2.09930635319\ i$
$e_4 = 6.89121600867$

and

$e_1 + e_2 + e_3 + e_4 = \underline{\mathbf{5.798972165}}$

MRM (Minimum Recommended Model) = HP-10C.

I'll post my original RPN solution for both the HP-10C *(3 steps)* and another more powerful Voyager model.

2: Simpler

Write a program to compute and output in one go all real roots of:

$$1^x + 19^x + 20^x + 51^x + 57^x + 80^x + 82^x = 2^x + 12^x + 31^x + 40^x + 69^x + 71^x + 85^x$$

accurate to at least 100 decimal places.

MRM: HP-42S. I'll post my original short solution for the HP-71B

3: Simple

Write a program to output all machine-representable real roots of the equation:

$$(x+1)^2 - (x+2)^2 - (x+3)^2 + (x+4)^2 - (x+5)^2 + (x+6)^2 + (x+7)^2 - (x+8)^2 = 0$$

where *machine-representable* means that the numerical value in question can be represented *exactly* in the finite precision of your particular model. For instance, **1E300** and **1.23456789012E-320** are machine-representable numbers in the HP-71B but they aren't in the HP-10C or HP-41CV, say. Your program must not ask for any input from the user and must output all the roots in ascending numerical order.

MRM: HP-11C. I'll post my original *2-line* solution for the HP-71B.

The Challenge:

1: Simplest

Write a program to split the set of integers [0,1, ... ,15] into two sets with the same number of elements such that the respective sums of the elements, the squares of the elements, and the cubes of the elements of each set are equal.

For instance, we can form the sets **[0,1,2,3,12,13,14,15]** and **[4,5,6,7,8,9,10,11]** and indeed both have the same number of elements (eight) and the same sums of the elements (60) but the sums of the squares are *unequal* (748 and 492), let alone the sums of the cubes, so this is hardly a solution.

MRM = HP-10C. I'll post my short, generalized RPN solution for the HP-10C as well as an even shorter generalized RPN solution *(9-step)* for a more powerful Voyager model

---

2: Simpler

The first notch of the recent ***"Valentine's Day 2008 Mini-Challenge"*** asked for a program to find five numbers when given their pairwise sums and it was stated there that for the general case of N numbers there's always a unique solution *except* when N is a power of 2, where multiple *distinct* solutions can be found. So you're asked to

Write a program to find two distinct non-negative sequences of 2^N elements which have the same set of pairwise sums

MRM = HP-10C. Again, I'll post my original RPN solutions for the HP-10C and a more powerful Voyager model, as well as a bonus version for the HP-71B.

3: Simple

Ms. Germain selects two integers between 1 and 100 (both excluded) and tells their sum only to Susan and their product only to Peter, both assumed to be ideally intelligent and honest students. She then asks if they can figure out what the original numbers are, and their dialogue goes as follows:

```
Peter:  I'm sorry Ms. Germain but I can't figure out the numbers right now ...
Susan:  Of course, I knew that you couldn't. Anyway, right now I can't either ...
Peter:  Thanks, Susan, because now I know them !
Susan:  You're welcome, Peter, for now I know them too !!
```

If they could, your HP can too, so:

Write a program to find the two original numbers.

MRM: HP-15C. I'll post my original solution for the HP-71B

4: Less Simple

Write a program to, given a set of integers, find a nonzero integer constant such that if you multiply each element by it, the set is now entirely composed of perfect powers.

The set may have up to five arbitrary (but distinct) nonzero elements, and an exact representation of the computed constant must be output.

For instance, if the given set is **[3,4]**, a possible exact solution is represented by the the constant **9**, because multiplying each element by it gives the set **[27,36]** and both **27** ($=3^3$) and **36** ($=6^2$) happen to be perfect powers.

Once written, use your program to find such a constant for these sets:

- The **Generations** set : [41, 67]
- The **USA** set : [1776, 2008]
- The **Then & Now** set : [2007, 2008]
- The **Halphacentaury** set : [1958, 2008, 50]
- The **Progressions** sets : [2, 3, 4] , [3, 4, 5] , [2, 3, 4, 5] and [2, 3, 4, 5, 6]

```
- The Voyager set        :  [ 10, 11, 12, 15, 16 ]
- The Excelsior set      :  [ 71, 42, 15 ]
```

MRM: HP-41CV. I'll post my original solution for the HP-71B

5: Least Simple

The sequence defined by:

$$(n+1)*x_{n+1} = n*x_n + x_n^k$$

when **k = 2** and $\mathbf{x_1 = 3}$ continues with

$x_2 = 6$, $x_3 = 16$, $x_4 = 76$, $x_5 = 1216$, and $x_6 = 247456$,

all of which are nicely *integer*. Regrettably, $\mathbf{x_7 = 8747993810.2857}$, which is *not*. That said, you're asked to

Write a program which, given k and x_1, both integer and > 1 , finds and outputs both the index of the first element of the sequence which fails to be an integer, if any, as well as its nonzero fractional part.

For example, for the value $\mathbf{x_1 = 3}$ above, your program should output **7** (index of the first non-integer element) and **2/7 = 0.285714285714** (nonzero fractional part of the first non-integer element).

Once written, use your program with **k = 2** and the starting values $\mathbf{x_1}$ **= 15, 41, 42, 50, 67, 71, 1958, 1992, and 2008** to find and output the corresponding indexes and fractional parts of the respective first noninteger elements. And if you succeed, see if your program can cope with the cases **k = 2**, $\mathbf{x_1 = 99}$ or the simpler-looking (!) **k = 3**, $\mathbf{x_1 = 2}$.

MRM: HP42S. I'll post my generalized original solution for the HP-71B

That's all. I think you'll heartedly agree with me that this time everything's *really simple*, so don't hesitate to take your favorite HP calc and spend an enjoyable time solving each and every one of them. Within a week or so I'll post my original solutions plus comments, as usual.

And remember: ***Do-Not-Cheat***. No googling, no computer programs in C# or Excel, no bending of the rules. That would be lame. You don't need that. You can do it on your own, in spades. I *trust* you ! Enjoy !

Best regards from V.

Ad. 1- Simplest

*Message #2 Posted by **Nenad (Croatia)** on 2 Apr 2008, 5:30 a.m.,*
in response to message #1 by Valentin Albillo

The result (sum of eigenvalues) is the sum of main diagonal elements of the original matrix, as I recall from Linear Algebra courses (though I do not remember why).

3 steps solution: + + +

Re: Ad. 1- Simplest

*Message #3 Posted by **John Keith** on 2 Apr 2008, 9:11 p.m.,*
in response to message #2 by Nenad (Croatia)

On the 48GX and later, only one step:
<< TRACE >>
;-)

Re: Pre-Challenges #2-3

*Message #4 Posted by **Steve Perkins** on 2 Apr 2008, 3:58 p.m.,*
in response to message #1 by Valentin Albillo

Pre-Challenge #2

I believe I have determined what the real roots of the equation are.

I just wish I could prove mathematically that these are the only roots. Here is a program for the HP-25 to output them:

```
01 6
02 f pause
03 g x=0?
04 R/S
05 1
06 -
07 GTO 02
```

Pre-Challenge #3

I'm not so great with calculators anymore, but I can (usually) do some algebra. I noticed quickly that the X^2 terms would cancel out giving a linear equation. So I expanded the terms. I ended up concluding that the equation is satisfied for ALL real numbers.

This leaves me with a puzzle. How to write a program to satisfy the letter of the problem statement. "Write a program to output all machine-representable real roots of the equation:"

Can a program be written to (theoretically) output ALL machine-representable values?

Re: Challenges #4

*Message #5 Posted by **Steve Perkins** on 3 Apr 2008, 10:59 a.m.,*
in response to message #4 by Steve Perkins

```
>2 DESTROY ALL @ DIM A(5) @ INPUT "HOW MANY (2-5)";C
>4 DIM P(5) @ P(1)=1155 @ P(2)=770 @ P(3)=924 @ P(4)=1980 @ P(5)=2100
>6 FOR I=1 TO C @ DISP "A(";I;")"; @ INPUT A(I) @ NEXT I
>8 FOR I=1 TO C @ PRINT A(I);"^";P(I); @ IF I<C THEN PRINT "*";
>10 NEXT I @ PRINT @ END
```

This is brute force (but simple). I started with the 2 number case (call them A,B). I soon found that a multiplier of A^2 * B^3 always worked. When you multiply by A you got a perfect cube. When multiplied by B a perfect square.

Extending to 3 numbers (A,B,C) I needed A^15 * B^20 * C^24 to assure even 2nd, 3rd and 5th powers. Of course this often isn't the smallest possible multiplier that satisfies, but it does seem to work.

The output for the sets given is kind of boring (and not optimal):

```
41 ^ 1115 * 67 ^ 770
1776 ^ 1115 * 2008 ^ 770
2007 ^ 1115 * 2008 ^ 770
1958 ^ 1115 * 2008 ^ 770 * 50 ^ 924
2 ^ 1115 * 3 ^ 770 * 4 ^ 924
3 ^ 1115 * 4 ^ 770 * 5 ^ 924
2 ^ 1115 * 3 ^ 770 * 4 ^ 924 * 5 ^ 1980
2 ^ 1115 * 3 ^ 770 * 4 ^ 924 * 5 ^ 1980 * 6 ^ 2100
10 ^ 1115 * 11 ^ 770 * 12 ^ 924 * 15 ^ 1980 * 16 ^ 2100
71 ^ 1115 * 42 ^ 770 * 15 ^ 924
```

Edited: 3 Apr 2008, 11:23 a.m.

Re: Challenges #4

*Message #6 Posted by **Valentin Albillo** on 3 Apr 2008, 11:49 a.m.,*
in response to message #5 by Steve Perkins

Hi, Steve:

Steve posted:

> ***"The output for the sets given is kind of boring (and not optimal)"***
>
> First of all, thanks for your interest and for your very nice posted solution. At first glance, it seems to be correct but I agree with you that the output is certainly boring and calling it *"not optimal"* is a vaaaaast understatement and could easily win a dozen prizes for "political correctness" :-)
>
> For instance, for the sample set **[3,4]**, where I stated that the single-digit constant **9** would do the work, your solution produces instead:

$$3^{1155} * 4^{770} =$$

```
 458397799047057184133972645862312531389
749843019498480466199744309993150702568 1
779896411251466456713225745888845430689 7
002619085678965680477957707166530556013 9
943940315588027507276860173430999580281 8
452032246525338139570315999540174140173 6
641541790349719356263087814667536271832 5
827664172314885835563590950845298058582 0
912358516515473496281134967087221593401 0
895853920048375998489068885977416722359 7
969345933865760146895070306637913705508 6
391536802736553069312886728156063752480 7
222613387059038936424475219936246075299 9
109043418666979979134128190553952641173 5
374292881793464433509728826160041280553 8
```

```
0676812261364454626098243479923850627181
3261947308737780205819640303201318473252
0645919638419607491890339252868846722906
0278437601589417584474744024781066878990
3044846282795871395522812107253913512313
0879958677258586602301016784867415278319
2671755248861308275282811073899411891547
2116799137036398413309711236546910379821
4874545869436763742826250051290255694699
2106935168238391718445225699447748748175
1030068183826432
```

which certainly could use a little "optimization". For the challenge sets the situation is much, much worse.

So, congratulations for giving such a short, clever solution (which I assume is fully correct, can't run right now), you've certainly done your part and fully deserve the credit, but this certainly wasn't what I had in mind.

I'm not asking for the smallest possible constant (though that would be nice) but at least something not very far from it, say a few orders of magnitude at most. Got what it takes ? :-)

Best regards from V.

Re: Challenges #4

*Message #7 Posted by **Egan Ford** on 3 Apr 2008, 11:58 a.m.,*
in response to message #6 by Valentin Albillo

The original problem states "exact representation". What does that mean exactly? Is a^x * b^y * ... allowed?

My solution (still working on it) provides the smallest constant. Some of the constants are 11 digits long. This would make the MRM of a 41CV a bit challenging.

Edited: 3 Apr 2008, 12:02 p.m.

Re: Challenges #4

*Message #8 Posted by **Valentin Albillo** on 3 Apr 2008, 12:06 p.m.,*
in response to message #7 by Egan Ford

Hi, Egan:

Egan asked:

> *"The original problem states "exact representation". Is a^x * b^y * ... allowed?"*

Yes, it is allowed. That's precisely why I used the wording *"exact representation"*. The expression used by Steve is an explicit, mathematically exact representation of the constant, even if it can't be evaluated in the particular machine's finite precision.

Though I would recommend that if said representation can be exactly evaluated without precision loss, it should be for the user's benefit.

Best regards from V.

Re: Challenges #4

*Message #9 Posted by **Steve Perkins** on 10 Apr 2008, 7:43 p.m.,*
in response to message #6 by Valentin Albillo

I haven't made as much headway as I hoped to on this problem. But it's not too hard to do better than I did.

```
10 DESTROY ALL @ DIM A(5) @ DIM P(5) @ INPUT "HOW MANY (2-5)";C
12 IF C=2 THEN P(1)=2 @ P(2)=3
14 IF C=3 THEN P(1)=15 @ P(2)=20 @ P(3)=24
16 IF C=4 THEN P(1)=105 @ P(2)=140 @ P(3)=84 @ P(4)=90
18 IF C=5 THEN P(1)=1115 @ P(2)=770 @ P(3)=924 @ P(4)=1980 @ P(5)=2100
20 FOR I=1 TO C @ DISP "A(";I;")"; @ INPUT A(I) @ NEXT I
30 FOR I=1 TO C @ PRINT A(I);"^";P(I); @ IF I<C THEN PRINT "*";
40 NEXT I @ PRINT @ END


 41 ^ 2 * 67 ^ 3
 1776 ^ 2 * 2008 ^ 3
 2007 ^ 2 * 2008 ^ 3
 1958 ^ 15 * 2008 ^ 20 * 50 ^ 24
 2 ^ 15 * 3 ^ 20 * 4 ^ 24
 3 ^ 15 * 4 ^ 20 * 5 ^ 24
 2 ^ 105 * 3 ^ 140 * 4 ^ 84 * 5 ^ 90
 2 ^ 1115 * 3 ^ 770 * 4 ^ 924 * 5 ^ 1980 * 6 ^ 2100
 10 ^ 1115 * 11 ^ 770 * 12 ^ 924 * 15 ^ 1980 * 16 ^ 2100
 71 ^ 15 * 42 ^ 20 * 15 ^ 24
```

Again, not optimal. Probably not close enough in many cases. But maybe it gives some insight to the problem.

I can see that getting the prime factorization of the numbers and looking for existing squares, cubes, or higher powers of primes can lead to optimization of the ultimate product. Also having factors in common can help. I can even see how to do it in the case of 2 numbers input. So far, more than 2 numbers have proven to be a challenging problem for me. I'll keep thinking about it.

Thanks for the enjoyable mental (and calculator) workouts, by the way. I have been looking forward to your challenges and have not been disappointed. I just wish I had the time (and brains) to give these the attention they deserve!

Re: Exercices 2&3

*Message #10 Posted by **J-F Garnier** on 2 Apr 2008, 4:02 p.m.,*
in response to message #1 by Valentin Albillo

As for the exercices 2&3, with the help of my HP-71B:

```
#2
20 FOR X=0 TO 259
30 L=1^X+19^X+20^X+51^X+57^X+80^X+82^X
40 R=2^X+12^X+31^X+40^X+69^X+71^X+85^X
50 IF R=L THEN DISP X
60 NEXT X
>RUN
 0
 1
 2
 3
 4
```

```
 5
 6


#3
Thanks to my 49G+ that showed me that the equation is identical to 0=0:
130 FOR E=499 TO -499 STEP -1
140 FOR M=9.99999999999 TO 1 STEP -1.E-11
150 DISP -M*10^E
160 NEXT M
170 NEXT E
180 DISP 0
190 FOR E=-499 TO 499
200 FOR M=1 TO 9.99999999999 STEP 1.E-11
210 DISP M*10^E
220 NEXT M
230 NEXT E


Or even simpler with the Math ROM (HP-71B specific):
260 X=-MAXREAL
270 DISP X @ X=NEIGHBOR(X,MAXREAL) @ GOTO 270
```

Re: Exercices 2&3

*Message #11 Posted by **Steve Perkins** on 10 Apr 2008, 7:56 p.m.,*
in response to message #10 by J-F Garnier

I like your solutions to these!

My question is, can we know that there are no more than 7 real roots to the equation given?

In the general case, it seems we can't even know that all real roots are integers, as I originally assumed. Consider this simpler example:

```
2^x + 32^x = 8^x + 18^x
```

How many real roots are there? What are they?

I have resisted searching the internet looking for the mathematical basis behind these types of equations. It does have me thinking, and that can't be a bad thing. :-)

SSMC20 Challenge 1.

*Message #12 Posted by **Egan Ford** on 2 Apr 2008, 4:54 p.m.,*
in response to message #1 by Valentin Albillo

> Quote:
>
> ***Write a program to split the set of integers [0,1, ... ,15] into two sets with the same number of elements such that the respective sums of the elements, the squares of the elements, and the cubes of the elements of each set are equal.***
>
> For instance, we can form the sets **[0,1,2,3,12,13,14,15]** and **[4,5,6,7,8,9,10,11]** and indeed both have the same number of elements (eight) and the same sums of the elements (60) but the sums of the squares are *unequal* (748 and 492), let alone the sums of the cubes, so this is hardly a solution.
>
> *MRM = HP-10C*. I'll post my short, generalized RPN solution for the HP-10C as well as an even shorter generalized RPN solution *(9-step)* for a more powerful Voyager model

I have two solutions for this problem. The first is brute force. The 2nd solution is based on my observations of the first solution.

Brute force HPGCC solution:

I started with the following identities that I obtained from one of my text books (*Excursions in Number Theory*, Pg 19):

```
Sum of sequential digits:          n*(n+1)/2
Sum of squared sequential digits:  n(n+1)(2n+1)/6
Sum of cubes sequential digits:    (n*(n+1)/2)^2
```

One of the sets would have a zero, so I only had to find 7 numbers out of 15 (6435 combinations total) that when summed all 3 different ways equaled 1/2 of the above equations with n = 15.

Output from my HPGCC/50g program (source at end of this post):

```
gotit!
iter=5279
0 3 5 6 9 10 12 15
and
1 2 4 7 8 11 13 14
Time to solution: 0 seconds
```

The first sequence of numbers all have an even number of 1s (base 2) and the second sequence all have an odd number of 1s.

My 16C program to calculate the sets:

```
001 -    42 34  CLEAR REG
002 -       24  DEC
003 - 43,22, 0  LBL 0
004 -    45 32  RCL I
005 -    43  7  #B
006 -        2  2
007 -    42  9  RMD
008 -    43 48  x!=0
009 -    22  1  GTO 1
010 -    45 32  RCL I
011 -    43 34  PSE
012 - 43,22, 1  LBL 1
013 -    43 24  ISZ
014 -    22  0  GTO 0
015 -    43 21  RTN
```

Toggling line 008 from x!=0 to x=0 will determine which set. After 8 numbers just kill the program.

I find this relationship astounding, but I have not had time to explore it much today. Perhaps this weekend. I did modify my brute force program to find more sets that meet the original problem criteria and discovered a few more things of interest:

1. The total number of elements must be k*8 where k > 1, i.e. 16, 24, 32, 40, etc...
2. The sets can be array base 0 or base 1, i.e., 0-15, 1-16, 0-31, 1-32, etc... all have solutions.
3. The solution for 1-16 is the same as 0-15 except that each element is one greater.

Here are the sets for k=3, 4, and 5 (array base 0):

```
k=3
gotit!
iter=619310
0 1 6 8 10 11 12 13 15 17 22 23
and
2 3 4 5 7 9 14 16 18 19 20 21


k=4
gotit!
iter=124340872
0 1 4 9 10 11 13 15 16 18 20 21 22 27 30 31
and
2 3 5 6 7 8 12 14 17 19 23 24 25 26 28 29


k=5
gotit!
iter=676322564
0 1 2 5 13 14 16 17 18 19 20 21 22 23 25 26 34 37 38 39
and
3 4 6 7 8 9 10 11 12 15 24 27 28 29 30 31 32 33 35 36
```

HPGCC Program:

```
#include <hpgcc49.h>
#define N 15


int a[15] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };
int Y1 = N * (N + 1) / 2 / 2;
int Y2 = N * (N + 1) * (2 * N + 1) / 6 / 2;
int Y3 = N * N * (N + 1) * (N + 1) / 4 / 2;
int done = 0;
int iter = 0;


void combos(int v[], int start, int n, int k, int maxk)
{
    int i, j;
    int x1, x2, x3;
    if (done)
        return;
    x1 = x2 = x3 = 0;
    if (k >= maxk) {
        iter++;
        for (i = 0; i < maxk; i++) {
            x1 += v[i]; x2 += v[i] * v[i]; x3 += v[i] * v[i] * v[i];
        }
        if (x1 == Y1 && x2 == Y2 && x3 == Y3) {
            printf("gotit!\niter=%d\n", iter);
            printf("0 ");
            for (i = 0; i < maxk; i++) {
                printf("%i ", v[i]);
            }
            printf("\nand\n");
            for (i = 0; i < n; i++) {
                int ok = 1;
                for (j = 0; j < maxk; j++)
                    if (a[i] == v[j]) {
```

```
                        ok = 0; break;
                    }
                if (ok) {
                    printf("%i ", a[i]);
                }
            }
            printf("\n"); done = 1;
        }
        return;
    }


    for (i = start; i < n; i++) {
        v[k] = a[i]; combos(v, i + 1, n, k + 1, maxk);
    }
}


int main()
{
    int v[7], start;


    clear_screen();
    cpu_setspeed(192 * 1000000);    // 192Mhz
    start = sys_RTC_seconds();
    combos(v, 0, 15, 0, 7);
    printf("Time to solution: %d seconds", sys_RTC_seconds() - start);
    SLOW_WAIT_CANCEL;
    return (0);
}
```

SSMC20 Challenge 2.

*Message #13 Posted by **Egan Ford** on 2 Apr 2008, 5:19 p.m.,*
in response to message #1 by Valentin Albillo

> Quote:
>
> The first notch of the recent ***"Valentine's Day 2008 Mini-Challenge"*** asked for a program to find five numbers when given their pairwise sums and it was stated there that for the general case of N numbers there's always a unique solution *except* when N is a power of 2, where multiple *distinct* solutions can be found. So you're asked to
>
> ***Write a program to find two distinct non-negative sequences of 2^N elements which have the same set of pairwise sums***
>
> *MRM = HP-10C*. Again, I'll post my original RPN solutions for the HP-10C and a more powerful Voyager model, as well as a bonus version for the HP-71B.

Hmmm... Same as challenge #1:

16C:

```
001 -    42 34  CLEAR REG
002 -       24  DEC
003 - 43,22, 0  LBL 0
004 -    45 32  RCL I
005 -    43  7  #B
```

```
006 -        2  2
007 -    42  9  RMD
008 -    43 48  x!=0
009 -    22  1  GTO 1
010 -    45 32  RCL I
011 -    43 34  PSE
012 - 43,22, 1  LBL 1
013 -    43 24  ISZ
014 -    22  0  GTO 0
015 -    43 21  RTN
```

Just kill after getting 2^N digits, then change 008 to x=0 and rerun to get the other set.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #14 Posted by **PeterP** on 6 Apr 2008, 1:09 a.m.,*
in response to message #1 by Valentin Albillo

I don't know how I missed this over the last couple of days! Thanks for a wonderful exercise again Valentin!

Anyway, I started at the end and while this is a challenge it is always an impossibility to me, so why not post some thoughts and ask for advice? (I have refrained from reading the existing posts so please forgive me if there is already a tip/solution for below. The headers make believe there is not, but who knows)

Looking at the last problem, it seems that (at least for 41's) the main problem would be how to handle the very quickly very large numbers. So lets definitely start with k=2 and ignore k>2 for the time being.

One way to deal with large numbers is to store them as their prime-factorization. starting with the definition of X_{n+1} we can write it also as (I think)

$$(n+1)*X_{n+1} = X_n*(X_n+n)$$

Now we also know that we only need to calculate X_{n+1} if X_n is an integer, which means it is divisible by n.

So with P_{X_n} being the product of all prime factors of X_n/n we can also write

$$(n+1)*X_{n+1} = nP_{X_n}*(P_{X_n} + n)$$
$$= n^2P_{X_n}*(P_{X_n} + 1)$$

Okay, and here is where I am stuck (don't be surprised that it is this early, remember its me, the blundering amateur of the crowd!) How can I get the prime-factorization of and number y+1 if I know the prime factorization of y?

I know that y and (y+1) don't share any prime-factors but that's about all. Any hints on how to think about this would be greatly appreciated!

Assuming, that this can be solved, below is the simple outline for a program which should be possible to implement even on a 41, though we might have some storage restrictions there. The main calculation burden is on calculating the prime factorization of n (there are plenty good algorithms for this and n, most likely, will be small anyway before the sequence stops although I assume it might be large for X_1 = 99…)

The calculation of relatively large numbers should be possible via this factor representation. Assuming that we can factor (y+1) relatively quickly given the factorization of y we should also have manageable run-times.

The biggest problem for the 41 would memory for the prime-numbers and factor loadings as we probably should start by storing say the first 100 prime-numbers or so. No problem here for the excelsior model 71b.

But the sticky point is the factorization of y+1 given the factorization of y…

After that the below flows more or less automatically modulo the bugs or oversights as the below obviously has not been actually tested. Just to be clear – I do not consider this at all an entry to the challenge as it is no program whatsoever. It is however a though and question and any hints in which way to think from here will be much appreciated.

```
1) Calculate and store the first 100 primes in an array P1/register block. So P1(1)=2, P1(2)=3, P1(3)=5,etc
2) Reserve array F1/register block for prime factor representation of Xn.
Each element of F1 holds the multiplicity of that particular prime number.
E.g. for a starting value Xn = 42 we would have F1(1) = 1, F1(2) = 1, F1(4)=1 as 42=2*3*7
3) Calculate prime factorization of X1
4) LBL Start
   5) Calculate prime factorization of (n)
   6) From that calculate PXn
   7) calculate a =  PXn Mod (n+1).
   8) check if a=0
      9a) yes => Xn+1 is integer. GTO CalcXN+1
   9b) no: calculate factorization of PXn + 1 = Q Xn
   10) calculate b =  QXn Mod (n+1).
   11) check if b = 0
      12a) yes=> Xn+1 is integer. GTO CalcXN+1
   12b) calculate c = a*b
   13)check c Mod (n+1) = 0
      14a) yes=> Xn+1 is integer. GTO CalcXN+1
   14b) no: Xn+1 is not integer. GTO FOUND


15)*LBL CalcXN+1
   16) sum the values in the factor-representation array/matrix for
       PXn, Q X[sub]n and n.
       Subtract the factor representation of (n+1).
       This is now the new Xn+1
   17) N=N+1
18) GTO Start


19)*LBL Found
   20) Prompt ‘non integer for ‘ N+1
   21) Prompt ‘Fraction = ‘ c/N+1
22) END
```

Tomorrow I hope to spend some time working further backwards, but I fear already for my productivity next Week...

Thanks again!

Peter

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #15 Posted by **Valentin Albillo** on 7 Apr 2008, 5:47 a.m.,*
in response to message #14 by PeterP

Hi, PeterP:

PeterP wrote:

"I don't know how I missed this over the last couple of days! Thanks for a wonderful exercise again Valentin!"

You're welcome, and let me thank you for your interest and kind words towards my humble efforts, much appreciated.

"I have refrained from reading the existing posts so please forgive me if there is already a tip/solution for below. The headers make believe there is not, but who knows"

Your intuition is correct. Not only isn't this subchallenge already solved but no one dare touch it with a ten feet pole, you're the very first to have the guts.

"Looking at the last problem, it seems that (at least for 41's) the main problem would be how to handle the very quickly very large numbers."

Correct. That's the one and only problem because computing the terms is an extremely simple exercise in basic arithmetic. However, it isn't just the 41C the one model to be affected by this circumstance but *any* HP model. Having an arbitrary-size multiprecision library won't automatically save the day with this one.

"Okay, and here is where I am stuck [...] How can I get the prime-factorization of and number y+1 if I know the prime factorization of y?"[/italic]

Your approach is quite ingenious, congratulations, but regrettably, though knowing the factorization of N+1 or N-1 (even if partial) can be of great help to factorize N, it is by no means easy to implement. Matter of fact, it is much more complicated than the simple procedure required to solve this subchallenge, which can be implemented in a few dozen lines of HP-71B code and probably in less than one line worth of HP50g RPL statements.

For instance, in the simplest case of $\mathbf{x_1 = 15}$, we find that the subsequent terms are

$x_2 = 120,\ x_3 = 4880,\ \ldots$

but by the time we reach $\mathbf{x_7}$ we already have:

$x_7 = 1007147427956055094294326891398595927564873870 1120$

and the factorizations of $\mathbf{x_7}$, $\mathbf{x_7 + 1}$, and $\mathbf{x_7 - 1}$ are, respectively:

$x_7 = 2 * 2 * 2 * 2 * 2 * 2 * 5 * 19 * 61 * 199 * 257 * 1871 * 562673 * 4204807 * 119949240928703736556753$

$x_7 - 1 = 3 * 7 * 7 * 11 * 17 * 37 * 41 * 53 * 733 * 1193 * 14033 * 397151 * 60709531 * 15401576400636983$

$x_7 + 1 = 33433783 * 30123645534101094521500211070897837901 4087$

which, as you can see, do involve very large factors and have very, very little in common.

Thus, unless you can deal with these large values and factors, I think another approach will prove more fruitful.

Thanks again and

Best regards from V.

Edited: 7 Apr 2008, 8:25 a.m.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #16 Posted by **PeterP** on 7 Apr 2008, 4:01 p.m.,*
in response to message #15 by Valentin Albillo

thanks for your kind words and hints Valentin, I'll keep working at it...

Cheers

Peter

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #17 Posted by **Egan Ford** on 8 Apr 2008, 12:12 p.m.,*
in response to message #15 by Valentin Albillo

> Quote:
>
> Not only isn't this subchallenge already solved but no one dare touch it with a ten feet pole, you're the very first to have the guts.

I made a lazy attempt last week before I left on vacation. Knowing that it would be impossible I still wrote an arbitrary precision version. And, yes it failed, but you already knew that. But, all was not lost. I did discover that:

```
x1 = 3,10       index = 7      remainder = 2
x1 = 4,6,7,9    index = 17     remainder = 4
```

The 3 and 10 cases didn't need arbitrary precision, but 4,6,7,9 needed more than 17000 digits. This data provides me with multiple test cases for my analytical solution. I'll try to have something this week or next.

Problem #5

*Message #18 Posted by **Eamonn** on 9 Apr 2008, 2:47 a.m.,*
in response to message #17 by Egan Ford

Hi Valentin,

Thanks once again for yet another great S&SMC. I don't know where you come up with the challenges, but it can't be easy to make them so that they have just the right amount of difficulty that a pocket calculator is a good tool to help solve them.

Anyway, I see that there have not been a lot of responses to the more difficult challenges so far. This is most likely due to the difficulty level and not due to a lack of trying on the behalf of the MoHPC audience. At least it gives slowpokes like me a chance to work on a solution before one is posted.

I believe that I have a solution to problem 5. It seems to work on paper, but I haven't had time yet to write a calculator program that automates it. I plan to do this in the next 24 to 48 hours and post it when I do so. I'm hoping that you will postpone posting your solution until I (and others that I am sure are working on this problem) get a chance to finalize and post my solution.

The basic idea is to calculate x(n) mod b and use that to calculate x(n+1) mod b. If x(b) mod b is not 0 then the solution is that the index is b and the remainder is x(b) mod b. Working with x(n) mod b keeps the figures very manageable.

The problem was really how to calculate x(n+1) mod b from x(n) mod b, which is easier to state than to do. Once I figured out how to do that, the rest was straightforward. As I said, I have only tried this on paper, but it seems to work so far.

Best regards,

Eamonn

Re: Problem #5

*Message #19 Posted by **Egan Ford** on 9 Apr 2008, 3:28 a.m.,*
in response to message #18 by Eamonn

> Quote:
>
> The problem was really how to calculate x(n+1) mod b from x(n) mod b, which is easier to state than to do.

If you are trying to solve ax = b mod m, just set x = 1, and increment x until ax mod m = b.

Re: Problem #5

*Message #20 Posted by **Eamonn** on 9 Apr 2008, 12:01 p.m.,*
in response to message #19 by Egan Ford

Thanks Egan. That's basically what I ended up doing, except I started with x=0. It should give the same results I believe.

Best Regards,

Eamonn

Re: Problem #5

*Message #21 Posted by **Valentin Albillo** on 9 Apr 2008, 7:28 a.m.,*
in response to message #18 by Eamonn

Hi, Eamonn:

Eamonn posted:

> *"Thanks once again for yet another great S&SMC. I don't know where you come up with the challenges, but it can't be easy to make them so that they have just the right amount of difficulty that a pocket calculator is a good tool to help solve them."*
>
> Thanks a lot for your interest, really appreciated. And no, it isn't easy to come up with them challenges. Tuning them up for difficulty is particularly troublesome so that's why I usually include several subchallenges of various difficulties, expecting that everybody will find some suitable for their perceived abilities.
>
> *"Anyway, I see that there have not been a lot of responses to the more difficult challenges so far. This is most likely due to the difficulty level and not due to a lack of trying on the behalf of the MoHPC audience."*
>
> I'm not so sure. Certainly, sub #4 and sub #5 are designed to offer a good fight to the likes of Egan, J-F, or yourself among others, but the rest are much more affordable, they basically require only ingenuity, interest, and some research and trial effort. Regrettably, people seem nowadays to be too busy, or too lazy, or both, or they simply aren't that interested in the programming, the algorithms, or even the underlying math anymore.

Perhaps they prefer to lurk and just see the efforts of others or even just browse through my final solutions and comments, without bothering to work them out for themselves or even post some comments of their own. I can't be sure this is the case, but I think it is a pretty educated guess. Happens the same with fan publications: people subscribe and then expect to receive a magazine choke-full with interesting information ... created by others; they aren't interested in making the effort to create and submit something interesting themselves, they simply lurk and leech. It is less tiring. Then they're surprised and upset when said publication hits the deck for lack of contributions.

"At least it gives slowpokes like me a chance to work on a solution before one is posted [...] I plan to do this in the next 24 to 48 hours and post it when I do so. I'm hoping that you will postpone posting your solution until I (and others that I am sure are working on this problem) get a chance to finalize and post my solution."

Certainly. I'll postpone my posting the original solutions and comments till next Monday so you all will have a chance at it. That's about 120 extra hours. This is the last S&SMC for the foreseeable future, the series ends its run here, so try and make the most of it.

Again, thanks for your efforts, for your appreciation, and for taking the trouble to post so that I would know.

Best regards from V.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #22 Posted by **Valentin Albillo** on 9 Apr 2008, 7:01 a.m.,*
in response to message #17 by Egan Ford

Hi, Egan:

Egan posted:

"Knowing that it would be impossible I still wrote an arbitrary precision version. And, yes it failed, but you already knew that."

Thanks for your interest and willingness to contribute, much appreciated. This and subchallenge #4 are the kind of teasers explicitly concocted to offer some resistance to people with your considerable abilities, all easier tasks having been met quite successfully by you in the past. And yes, I knew.

"But, all was not lost. I did discover that [...] The 3 and 10 cases didn't need arbitrary precision, but 4,6,7,9 needed more than 17000 digits."

Your results are correct so far, though none of them included any of the asked cases. Remember that the last one is for k=3, not k=2 like all the rest, so your program should ideally cope with general integer k > 1 as well.

"This data provides me with multiple test cases for my analytical solution. I'll try to have something this week or next."

And I'll be eager to see what you come up with. **I will extend the period before I post my original solutions to next Monday** so that enough time is available for you to try and solve them.

Anyway, I must say that it's a pity that this S&SMC#20 has apparently failed to arouse the same kind of interest as the recent ones did: more than a week has elapsed and the three best subchallenges (#3, #4, #5) remain essentially unsolved, including #3 which hasn't even been mentioned by anyone. The interest for the others hasn't been that high either.

I was under the impression that, since a full year has elapsed since the previous S&SMC#19, challenge-loving, math-loving people would be eager to try their wits against my newest production, which is carefully crafted to be both suitably difficult (but not unassailable, mind you) as well as aesthetically pleasing, with some amazing identities, unexpected numeric

relationships, and awesome results and algorithms being brought up for public discussion and examination.

Regrettably it seems that either I missed the mark with the difficulty or else people are simply too busy and/or too lazy to allocate time and effort to study and solve them. Previous challenges would see a large number of varied solutions for all kinds of models, from large hand-crafted RPN programs to short, mysterious RPL routines, as well as some BASIC, C, or even FORTH implementations. Now we haven't seen more than a line of RPL code in all (that TRACE command) and no one has tried to create their own version after seeing other solutions, or even optimize the ones given (as could have been the case for training-#3 and subchallenge-#4).

I take this as a strong signal that my S&SMC's have overstayed their welcome and have now been relegated to the status of minimally interesting topic only appreciated by a very tiny minority (perhaps less than a handful people) while the vast majority of forum regulars and visitors aren't that interested anymore, if at all.

Since these challenges do take a lot of time and effort to concoct (research suitable topics, long-list and short-list the most awesome and better suited, create original solutions for them, format and post the very long challenge message, format and post the very long solutions & comments, etc), this is a book case of 'diminishing returns' in which I've reached the point where the small participation does not justify the large effort so this will be the last S&SMC.

After all, 20 is a nice round number so we can adequately call it quits after this one, don't you think ? :-)

Best regards from V.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #23 Posted by **Maximilian Hohmann** on 9 Apr 2008, 7:22 a.m.,*
in response to message #22 by Valentin Albillo

Hello!

Although it was not addressed at me, I MUST answer this one:

> Quote:
>
> After all, 20 is a nice round number so we can adequately call it quits after this one, don't you think ? :-)

No, no, no! 20 is not a nice round number. I am far too dumb to come up with a solution for even the "simplest" of your challenges (seriously considering of having my academic grade erased from my passport now :-( ) but as long as you keep coming up with new ones there is always hope!
I would suggest 314 as a nice round number for the final challenge :-)

Greetings, Max (looking forward to your solutions - even got myself a MATH ROM for my 71B since the last challenge, so that I can, at least, type them into the calculator myself)

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #24 Posted by **Valentin Albillo** on 21 Apr 2008, 10:42 a.m.,*
in response to message #23 by Maximilian Hohmann

Hi, Max:

Max posted:

> *"No, no, no! 20 is not a nice round number"*

Yes it is. It's been a number of years since I issued S&SMC#1 and the difficulty has been steadily increasing because of the incredible abilities of interested people like yourself, who quickly solved every challenge no matter how fiendishly tricky.

Logically, the time required to concoct, create original solutions, and format and post them has likewise increased, while at the same time participation has decreased to the point that only a handful of people dare to attack them, even when they actually aren't that difficult really.

See for example the training problem which essentially asked for a program to output every machine-representable number. This was allegedly interesting and challenging and not difficult at all; Jean-François provided a solution which could be easily ported to every machine out there and not specially difficult to discover from scratch. Yet apart from him, no one posted a thing.

This being so, I think it's time to end the series because of diminishing returns, and will stick instead to simpler mini-challenges for a while.

"Greetings, Max (looking forward to your solutions - even got myself a MATH ROM for my 71B since the last challenge, so that I can, at least, type them into the calculator myself)"

Thank you very much for your interest, first part of solutions posted; anyway, I would suggest that it's actually much easier and a better investment of time to use Emu71 instead and simply copy-paste the text from my solutions post to it, thus avoiding boring and error-prone tasks such as entering and checking out by hand a long listing.

Best regards from V.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #25 Posted by **J-F Garnier** on 9 Apr 2008, 7:33 a.m.,*
in response to message #22 by Valentin Albillo

Hi Valentin,

You wrote: *...this S&SMC#20 has apparently failed to arouse the same kind of interest as the recent ones did...*

I can only speak for myself, but I always had an high interest in your S&SMC. But maybe this time my math skills are not good enough to solve your last "Oh So Simple !..." challenges.

I found the problems #3 and #4 quite interesting for me and thought a lot about them, but was unable to devise a suitable program to solve them. I found some particular solutions "by hand" for #4, but nothing general.

However, as you allow us a few days more, I may spend some more time on it.

Thanks for your efforts in preparing these S&SMC and best regards,

J-F

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #26 Posted by **Valentin Albillo** on 21 Apr 2008, 10:44 a.m.,*
in response to message #25 by J-F Garnier

Hi, Jean-François:

You've done more than your share, and I appreciate it very much.

I look forward to see you participate in future mini-challenges, which will probably be centered around the HP-15C and the HP-71B, my two favorite models since ever. Thanks for your continued interest and appreciation, and

Best regards from V.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #27 Posted by **Egan Ford** on 9 Apr 2008, 2:35 p.m.,*
in response to message #22 by Valentin Albillo

Valentin,

I think it is a combination of many factors. Most are probably just busy. And, I think that this set of problems while fascinating is also difficult to get started.

I believe that your most successful challenges are simple to understand immediately. And while there is always something hidden and most fascinating in the details, the problems are simple enough for many to submit an initial good guess on a lunch break. This starts the ball rolling. And keeps the problem in their minds for follow up. Some of my favorites have been HP-15C Bits o' Pi, Vday 07 (last non-zero digit of n!), and palindromic square (42S).

I would consider relaxing the restriction on Google. Research is a real world exercise. I find discovery through the work of others just as rewarding as self-discovery. Research is essential to save time. This may solve the "busy" factor and allow others a way to get started. Its not like Google is going to write the program for you. There is nothing worse than giving up in frustration because there is no avenue to aid. With the exception of SSMC19 I do not think you have ever restricted Google. I'll admit for SSMC19 A+ I did Google (although I didn't have to, someone else also did and sent me a paper to read). But, if I had not had read that paper, I would have never in a lifetime figured out how to write a program for A+. It was a difficult paper to understand and convert to code. During the process I got to correspond with the author of the paper, got copies of his research, and learned a bit of German since I had to translate much of the research myself. Quiet rewarding and I would have missed out on all of this if I and others had not Googled. BTW, it still took me 40 hours to understand this paper and write the program.

I would not give up. Perhaps a series of mini-challenges is in order. I'll assume it is less work for you and probably much less work for others. If mini-challenges do not draw in the crowds, then perhaps your right and it is the end of an era.

I could be completely off-base here. Perhaps if your next challenge had the words "eBay" or "fantasy calculator" in it, then more would be interested.

Thank you for your efforts. I mostly visit for the challenges, it would be sad to see them go.

P.S. here is a "challenge" for you:

Write a program to find non-zero integers a, b, c, and d:

$a^4 + b^4 + c^4 + d^4 = (a + b + c + d)^4$

The spoiler is in AMM March 2008. I have not had the time to finish my program, but you should have no problem writing this in your sleep.

Edited: 9 Apr 2008, 3:34 p.m.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #28 Posted by **Valentin Albillo** on 9 Apr 2008, 4:34 p.m.,*
in response to message #27 by Egan Ford

Hi, Egan:

> Good advices, all of them. The reason why I've been refraining from palindromic squares and such is that, frankly, they seem quite trivial to me and do not have much mathematical meat, so to say.
>
> The fact that certain 12-digit number is the only palindromic square, say, it's a nice curiosity but nothing else. It goes nowhere and teaches you nothing mathematically speaking, though it can be a good programming exercise.
>
> On the other hand, the fact that some apparently integer sequence fails to produce an integer after 600 terms have been generated is not only fascinating but discovering why can teach you a lot about divisibility, congruences, and working in modular arithmetic to avoid having to deal with numbers so large that no computer would ever be able to even store them, let alone perform operations on them.
>
> Nevertheless you're probably right. The former kind of challenge attracts a larger number of people than the latter, despite its irrelevancy so I might heed your advice and stick to simple mini-challenges for the HP-15C and such for now.
>
> As for your 'challenge', I've read the same AMM issue as you did and know as much about the problem as you do, including that one possible solution is A = -2634, B = 955, C = 1770, and D = 5400, as can easily be checked:
>
> ```
> > A^4+B^4+C^4+D^4 , (A+B+C+D)^4
>
> 909087685468561         909087685468561
> ```
>
> Writing an elliptic-curve-based program to find some of the infinite solutions is hardly appealing to me right now. I'd rather use the elliptic curve approach to try and write a good factoring program instead.

Thanks and best regards from V.

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #29 Posted by **PeterP** on 12 Apr 2008, 10:29 p.m.,*
in response to message #22 by Valentin Albillo

Dear Valentin,

for what its worth let me say that your post deeply saddened me. I personally was very much looking forward to this challenge, having missed due to workload the valentine's day challenge.

Rest assured that I follow each and every challenge posted here with great interest and follow links posted and solutions shown. I have learned a great deal over the last few years through the excellent quality of the challenges that you post and the immense ingenuity of those who participate and solve them.

I would believe that there is a nice crowd of silent followers who greatly enjoy your challenges and the thread of responses. At the same token, let me tell you from the perspective of a complete amateur that it takes a certain amount of guts to openly participate given the high level caliber of your challenges and the usual contributers (Egan, Eammon, J-F, Karl, Max, etc). It is always a very humbling experience for me to toil for days over individual problems and then see beautiful and simple solutions posted by others within mere minutes or a few hours after you posted the challenge!

Yet for me this is an enormously enriching experience and - quite frankly - almost the only time I program these calculators. It gives me a reason to think, try and learn and find out more about these wonderful calculators.

Lastly I'd like to point out the obvious - it is very hard for you 'see'/gauge all the busy work and engagement that is going on behind the scenes before people post. From your perspective it must be hard to tell if a challenge is ignored or feverishly worked upon. Personally, I have looked into and exercised 'recreational math' with almost all problems but not always found a solution within the time or the time to find a solution.

While I can most definitely appreciate that time is a precious resource for you, FWIW, let me throw my vote in the hat for keeping your challenges a regular part of this forum. I would be deeply saddened to see them go.

Cheers

Peter

Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special "Oh So Simple !..."

*Message #30 Posted by **Valentin Albillo** on 21 Apr 2008, 11:12 a.m.,*
in response to message #29 by PeterP

Hi, Peter:

Peter posted:

> *"for what its worth let me say that your post deeply saddened me."*
>
> Infinite thanks for your unfailing interest, Peter, but no need to feel sad. S&SMC's were growing extremely difficult for most people, and it's preferable to end the series now and go back to the origins for a while, with even shorter, much simpler mini-challenges.
>
> *"I have learned a great deal over the last few years through the excellent quality of the challenges that you post and the immense ingenuity of those who participate and solve them."*
>
> Yes, I agree that you do have a point here. I've always been amazed at the extreme ingenuity displayed by some people when attacking my challenges. It's forced me to go to ever higher difficulty levels and still they would cope. I'm awed.
>
> *"I would believe that there is a nice crowd of silent followers who greatly enjoy your challenges and the thread of responses."*
>
> I concur they might exist, like fairies, but Occam's razor applies here and the simplest explanation is the one to choose: between a *'nice crowd of silent followers'* and '... everything's silent simply because there's no one ...', the latter is the logically simpler and thus preferable.
>
> *"Yet for me this is an enormously enriching experience [...] It gives me a reason to think, try and learn and find out more about these wonderful calculators."*
>
> Thanks a lot for your uplifting words, that was exactly my idea: to refine the art of programming HP calcs to the utmost, to discover beauty and awe in unusual mathematical facts, and to combine both for the better enjoyment and 'glory' of these wonderful machines we all admire and feel proud of. But ...
>
> *"From your perspective it must be hard to tell if a challenge is ignored or feverishly worked upon."*

Yes it is; that's the problem. I don't like to indulge in wishful thinking so if I get silence, I assume there's no one there and move on. Otherwise I'd feel as if I was deluding myself, and there's no need for that, I've got many other things to do. 'Preaching in the desert' wasn't made for me, I only like to 'preach' or 'teach' to interested, existing audiences :-)

"let me throw my vote in the hat for keeping your challenges a regular part of this forum. I would be deeply saddened to see them go."

Thanks again, again much appreciated. Don't worry, the challenges will remain, but in an easier, simpler format, as much more affordable Mini-challenges.

As for the regular S&SMC, if you're a Star Trek fan, like me, you'll certainly remember the title of the Next Generation series finale:

"All Good Things ..."

Best regards from V.

Short & Sweet Math Challenges

Message #31 Posted by ***Karl Schneider*** *on 20 Apr 2008, 4:37 p.m.,*
in response to message #22 by Valentin Albillo

Quote:

Regrettably it seems that either I missed the mark with the difficulty or else people are simply too busy and/or too lazy to allocate time and effort to study and solve them....

Since these challenges do take a lot of time and effort to concoct ... this is a book case of 'diminishing returns' in which I've reached the point where the small participation does not justify the large effort so this will be the last S&SMC.

After all, 20 is a nice round number so we can adequately call it quits after this one, don't you think ? :-)

Valentin --

I certainly can appreciate the time and effort you put into developing and preparing these challenges. It seems to dwarf even the substantial investments I've put into some of my own contributions (e.g., the analysis of your 7x7 "Albillo Matrix #7" and Rodger Rosenbaum's counter-example).

I'll admit, however, that I haven't seriously tackled very many of your challenges, because I don't find them particularly easy or simple. I don't really have the "knack" for the kind of programming required, or the particular math knowledge (e.g., number theory), to make these challenges a straightforward exercise. I'm continually astounded, though, how certain people have always been able to post working solutions so quickly after you post the challenge -- oftentimes before I've even had a chance to see it.

And yes, *time* is certainly the issue for me and others. There are many things that need to be done, but which would *not* get done if I/we invested free time in recreational solving of math problems. If such a challenge is difficult, I for one might not have enough mental effort to give to it, after the workday is over.

My contributions in the MoHPC Forum tend to address known practical issues of HP calculators: identify the specific problem, and provide a concise assessment of its root cause.

A few recent examples:

HP-35s slowness with integration

HP-15C bug: CHS and Stack Lift with zero

HP-33s/35s trigonometric bug tabulated

As you can see, it's kind of "inside-the-box, A-B-C" thinking. It's reasonably gratifying to find the definite and indisputable answer to a given problem without entering an ongoing process and giving extensive effort. (If only the day job were like that!) :-)

No one should complain if you decided to "call it day" regarding your S&SMC's. 20 of these is a very impressive body of work in itself (along with your may other contributions). A short article containing links to your issuance and solutions to each challenge in the Forum Archives would be a welcome addition to the Articles Forum. In fact, someone a while ago had undertaken a similar task to provide a list of challenges to date.

Best regards,

Karl S.

Edited: 21 Apr 2008, 3:10 a.m.

Re: Short & Sweet Math Challenges

*Message #32 Posted by **Valentin Albillo** on 21 Apr 2008, 11:55 a.m.,*
in response to message #31 by Karl Schneider

Hi, Karl:

Karl posted:

> *"I certainly can appreciate the time and effort you put into developing and preparing these challenges."*
>
> Thanks a lot, I know you do because it takes writing lengthy articles and analysis to get to fully understand just how much effort and time are required, as you've done many times in the past.
>
> *I'm continually astounded, though, how certain people have always been able to post working solutions so quickly after you post the challenge -- oftentimes before I've even had a chance to see it.*
>
> Same here. I used to comment with my daughter about how tricky and delightfully mischievous some S&SMC was, before I posted it, to later have to confess that a number of people saw through it all as if it were a Math 101, rookie-like problem, to her general amusement.
>
> *"There are many things that need to be done, but which would not get done if I/we invested free time in recreational solving of math problems."*
>
> Same here as well. But nevertheless, I would gladly obligue, even sacrificing sleep or other activities, if only I felt that there was a sizable demand or interest for them. But fact is there isn't and so I can't justify the effort to myself any longer. Sad, but a fact nevertheless.

"My contributions in the MoHPC Forum tend to address known practical issues of HP calculators: identify the specific problem, and provide a concise assessment of its root cause."

> Yes, and we're all the more grateful to you for it. In the long run, your contributions are far more permanent and of general interest than my admittedly ludic ones.

"No one should complain if you decided to "call it day" regarding your S&SMC's. 20 of these is a very impressive body of work in itself (along with your may other contributions)."

> Thanks. S&SMC#20 is definitely the 'series finale', with much simpler and shorter mini-challenges taking its place occasionally.

"A short article containing links to your issuance and solutions to each challenge in the Forum Archives would be a welcome addition to the Articles Forum. In fact, someone a while ago had undertaken a similar task to provide a list of challenges to date."

> I don't remember who right now. I don't do articles for the MoHP forum (nor for anyone else for that matter) but perhaps someone will do what you say, eventually. Nevertheless all my Short & Sweet Math Challenges and Mini-Challenges' threads can be freely downloaded in PDF format from my own calc site, just google for *"albillo"* and click the *second* link.

Thanks for your useful comments and

Best regards from V.

Re: Challenge 1

*Message #33 Posted by **PeterP** on 7 Apr 2008, 4:01 p.m.,*
in response to message #1 by Valentin Albillo

First problem – the below is a solution to the first problem for the 41CV. It is not completely brute force but clearly uses a lot more brawn than brain (of the 41 that is. the small build user has neither in sufficient quantity…)

It uses two ideas. First, to get the sum of the two octets to match we shall create pairs of equal sum and can resort to only creating quartets of those pairs and look for those that have equal sum of their squares and cubes. This is similar to the idea that Gauss used to derive the formula for the sum of n integers.

Second, it is obvious that the sum of squares and sum of cubes we are looking for in our octet is exactly halve of the sum of the squares/cubes of all integers.

The structure of the behemoth below is as follows

```
(1)  Pre-fill the registers with the sum of the squares and cubes of  the pairs in a convenient fashion
     so that indirect register polling can grab the right contents
(2)  It then uses flags to keep track of which pairs it has selected
(3)  It has 4 nested loops to run through all unique combinations roughly starting with the highest possible sum.
     When it finds a sum that is equal to ½ of the total, it goes to check also the cubes, which, as it turns out, also match at this  point.
     I shall think more about this 'coincidence'…
     However, if they were not to match, it would keep on going down the list if possible combinations.
     Lastly, it does not take advantage of the fact that after a certain point
     all options will be < than ½ the total and we could jump directly to 'No solution possible'
(4) Regs 0 – 7 hold the sum of the square of the relevant pair. (0 for (15,0), 1 for (14,2), etc
```

(5) Regs 10 – 17 hold the sum of the cubes of the relevant pair
(6) Reg 08 is the general loop reg for loops over the 8 paris
(7) Reg 20, N,O,M are the loop regs for the search loop over unique combinations
(8) Reg 18 (:=0.007), 09 (:=10), 23(:=sum of all cubes), 24(:=7), 25(:=sum of all squares) are just placeholders
    as the slowest instructions on the 41 are numbers so you want to store them in registers if you use them often

Run-time is actually not bad at about 55sec on a physical standard speed hp41, including all the feedback that is built into the code.

Its not pretty, but it works…

BTW - I wish Egan had finished his article about the 41c emulators before the challenge!
If his excellent article about the Voyager is any hint on its quality, I would have saved myself a lot of time typing the code
into here (Just trying to say that I thought it to be an excellent
article and I am very much looking forward to the version about the 41!)

```
LBL ‘Loop’
XEQ ‘STOSQ’
‘Start Search’
AVIEW
CLX
X<>F
CLA
RCL 18 (:=0.007)
STO 20
LBL ‘20
RCL 20
1+
STO N
LBL ‘N
    RCL N
    1+
     STO O
     LBL ‘O
           RCL 24 (:=7)
           RCL O
            1+
           X>Y?
                 GTO ‘NS
            STO M
            LBL M
                   CLX
                   X<>F
                   SF IND 20
                   SF IND N
                   SF IND O
                    SF IND M
                     XEQ C
                     VIEW X
                     RCL 25
                     X=Y?
                            GTO E
                       ISG M
             GTO M
             LBL ‘NS
             ISG O
     GTO O
```

```
      ISG N
GTO N
‘No Solution
TONE 0
AVIEW
RTN
GTO ‘LOOP


LBL C ‘calculate the sum of squares for quartet
      RCL IND 20
      RCL IND N
      +
       RCL IND O
       +
        RCL IND M
        +
RTN


LBL E    ‘check if sum of cubes also matching
      TONE 5
      TONE 9
      TONE 5
      CLX
       STO 26
       RCL 18 (:=0.007)
       STO 08
       LBL 12
              FS? IND 08
                     XEQ F
               ISG 08
        GTO 12
        CLX
        X<>26
        VIEW X
         RCL 23
         X=Y?
               GTO 99
RTN
LBL F    ‘get the right sum of cubes
     RCL 08
     RCL 09 (:=10)
      +
      RCL IND X
      ST+ 26
RTN


LBL 99  ‘great, we found it!
      BEEP
      “FOUND
      AVIEW
       CLA
       RCL 18 (:=0.007)
       STO 08
       LBL 14
              FS? IND 08
                      XEQ 15
               ISG 08
```

```
        GTO 14
         AVIEW
         STOP
GTO ‘Loop
LBL 15 ‘create result string in display
       XEQ IND 08
       ARCLI
       ‘k,        k = append sign
       RDN
       ARCLI
       ‘k,
RTN


LBL ‘STOSQ ‘fill the registers with sum of squares, cubes etc
      ‘Fill Regs…
       AVIEW
       CLX
       STO23
       STO 25
       STO 26
       7
       STO 24
       10
       STO 09
       0.007
       STO 08
       STO 18
       LBL 10
             XEQ IND 08
             XEQ 23
             STO IND 08
             ST+25
             CLX
             RCL08
             10
             +
             X<>Y
             STO IND Y
             ST+ 23
             ISG 08
       GTO 10
       2
       ST/ 25
       ST/ 23
RTN
LBL 23 ‘calc x^2 + y^2 (into X) and x^3+y^3(into y)
      Enter
       X^2
       St* Y
       RCL Z
       X^2
       ST* T
       +
       RDN
       +
       R^
RTN
```

```
LBL 00
15
0
RTN
LBL 01
14
1
RTN
LBL 02
13
2
RTN
etc…


LBL  07
8
7
RTN
```

Edited: 7 Apr 2008, 4:13 p.m.

Re: Challenge 1

*Message #34 Posted by **Egan Ford** on 8 Apr 2008, 11:47 a.m.,*
in response to message #33 by PeterP

> Quote:
>
> BTW - I wish Egan had finished his article about the 41c emulators before the challenge! If his excellent article about the Voyager is any hint on its quality, I would have saved myself a lot of time typing the code into here (Just trying to say that I thought it to be an excellent article and I am very much looking forward to the version about the 41!)

I have been swamped with professional and personal obligations. My goal for all 4 parts to be complete by 4/1 slipped.

My 41CX article briefly discusses HP41UC for compiling and printing. The proper HP41UC documentation is here: http://www.hpmuseum.org/software/41uc.htm. The latest version can be obtained from http://hp41.claughan.com/files/hp41uc.zip . It is a DOS program. If you are a Mac or Linux user then you can install DOSBox to run the utility. This is covered in my article. As for emulators, so far I have only documented V41 and EMU41. EMU41 is DOS-based, so if you got HP41UC working you have EMU41 working too. That is the easy part. The hard part is documenting all the different methods of getting code to and from a physical 41 for Windows, Mac, and Linux. I've already done a lot of the research, but there is still so much more to do. Perhaps I'll just skip that part.

Re: Challenge 1

*Message #35 Posted by **PeterP** on 8 Apr 2008, 12:09 p.m.,*
in response to message #34 by Egan Ford

Egan,

Thanks for the kind info. I have been looking into SDK41 and I am working on converting the original txt based documentation into a nicer formatted pdf document and maybe some extra info. Also, though I am lightyears behind your level of understanding, I'd be happy to collaborate and write up/research some parts so that you might have some basis to work with rather than having to start from scratch. Just a thought, you know how to reach me :-)

Cheers

Peter

Re: Challenge 1

*Message #36 Posted by **Egan Ford** on 8 Apr 2008, 12:29 p.m.,*
in response to message #35 by PeterP

> Quote:
>
> Also, though I am lightyears behind your level of understanding

I wouldn't say that. I've been a 41CX *user* for less than a year. As I discover things that are not well documented or fragmented in too many places I try to make a point of documenting it for other 41 n00bs. Kind of a cookbook, start to end.

> Quote:
>
> I'd be happy to collaborate and write up/research some parts so that you might have some basis to work with rather than having to start from scratch. Just a thought, you know how to reach me :-)

Thanks, that'd be great. I'll send you a draft in a few weeks for your review.

Challenge 2

*Message #37 Posted by **PeterP** on 8 Apr 2008, 12:18 p.m.,*
in response to message #1 by Valentin Albillo

First, let me admit that I did read the thread for the recent Valentine's day challenge and shamelessly used some information left there by Valentin. I hope that I can get a 'by' for this cheating from the kind author…

I started this second challenge by looking at known solution to see if there is anything to learn from them

For n = 2 we have (posted by Valentin in the Valentine Day's challenge) and n = 3 (from my solution for the first challenge of SSMC20)

```
|1  2  4  7|
|0  3  5  6|
|1  2  4  7  8  11  13  14|
|0  3  5  6  9  10  12  15|
```

Two observations:

- The first 4 numbers are the same for n=2 and n=3
- The absolute distance between elements of the first sequence and the second sequence at the same place is always 1.

Based on the above observation I thought about creating the solution for n=4 by trial and error and while doing so I made a third observation

- There seems to be a reversing pattern in how the sequence is created.

Let me explain what I mean by this. Lets only look at pairs of numbers at position x for the two sequences and lets call it

```
(u,l)
```

for upper and lower sequence. Additionally, lets denote with a '1' if the upper sequence has the higher number and with a '0' if the lower sequence as the higher number. This representation yields the following string of 1s and 0s for n=2,3,4 where I have taken the liberty to delineate blocks of length n-1.

```
|10|01|
|1001|0110|
|10010110|01101001|
```

This is a neat reversing patter where we can write an explicit formula for the 1's and 0's as a not() function of previous 1's and 0's as per below with x(i) being the value at position i

```
i = 1 to N;
x(2^(i-1)+1 to 2^i) = NOT(x(1 to 2^(i-1))
```

So lets just start with u(1) = 1 and l(1) = 0. This means that x(1) = 1 as the upper sequence has the higher number. Recursively using the above formula we get

```
x(2)    = not(x(1))   = 0
x(3-4)  = not(x(1-2)  = 01
x(5-8)  = not(x(1-4)  = 0110
x(9-16) = not(x(1-8)  = 01101001
```

which in turn delivers the recipe and solutions for n = 1,2,3,4 as

```
|1  0| => |1 2|
          |0 3|
|1  0  0  1| => |1  2  4  7|
                |0  3  5  6|
|1  0  0  1|0  1  1  0| =>|1  2  4  7  8  11  13  14|
                          |0  3  5  6  9  10  12  15|
|1  0  0  1|0  1  1  0|0  1  1  0|1  0  0  1| =>|1  2  4  7  8  11  13  14  17  18  20  23  24  27  29  30|
                                                |0  3  5  6  9  10  12  15  16  19  21  22  25  26  28  31|
```

From here it is relatively straight forward to write a program, however from a quick glance at the manual, it seems that the HP16c (which I have never used) would be most suited given his extensive capabilities with binary numbers. The below code is an implementation for the 41 and is generally very inefficient as it uses 1 reg per position so it maxes out at about n=7. If one would want to make it work for larger numbers, one could devise a space saving mechanism to store up to 31 positions in each register and use the Bit? function of the AdvModule (or somewhat more convenient functions of the CCDmodule). However, I would assume that someone familiar with the HP16c can write a much shorter and faster program…

The code below creates first the sequence of 1's and 0's by a simple double loop flipping each bit separately before it proceeds to use the bit-pattern to create the sequence. Output is in the form of (u,l) and sequential. Keep pressing r/s for each new pair to appear. As usually there is a bunch of STO functions at the start for repeatedly used numbers as they are so sloooow on the HP41.

```
LBL 'C2
CLRG
'N?
PROMPT
STO 00
2
STO 05
X<>Y
Y^x
12
+
PSIZE
-1
STO 03
ABS
```

```
STO 04
10
STO 06
+
STO 07
RCL 00
0.001
STO 01
*
RCL 04 (=1)
+
STO M
LBL 00 'loop over i = 1 to N
  RCL 05 (=2)
  RCL M
  INT
  RCL 04 (=1)
  -
  Y^X
  STO O
  RCL 04 (=1)
  +
  RCL 06 (=10)
  +
  RCL 05 (=2)
  RCL M
  INT
  Y^X
  RCL 06 (=10)
  +
  RCL 01 (=0.001)
  *
  +
  STO N
  LBL 01 'inner loop over all elements 2^(i-1) +1 to 2^i
    RCL N
    RCL O
    -
    RCL IND X
    RCL 03 (= -1)
    +
    ABS
    STO IND N
    ISG N
  GTO 01
  ISG M
GTO 00
LBL 02 'display loop
  RCL 04 (1)
  STO 11
  RCL 07 (=11)
  RCL 05 (=2)
  RCL 00 (=N)
  Y^X
  RCL 06 (=10)
  +
  RCL 01 (=0.001)
  *
  +
  STO 02
```

```
  CLx
  Enter
  RCL 04 (=1)
  LBL 03 ‘loop to display one pair
    RCL IND 02
    X=0?
    XEQ 05
    RDN
    ‘<
    ARCLI
    ‘k,
    X<>Y
    ARCLI
    ‘k>
    AVIEW
    STOP
    X>Y?
    X<>Y
    RCL 05 (=2)
    +
    RCL 04 (=1)
    X<>Y
    +
    LastX
    ISG 02
  GTO 03
BEEP
STOP
GTO ‘C2
LBL 05
  RTN
  X<>Y
  R^
RTN
```

SSMC20 Challenge 5. (answers w/ question)

*Message #38 Posted by **Egan Ford** on 9 Apr 2008, 3:08 a.m.,*
in response to message #1 by Valentin Albillo

Valentin,

can you confirm if the following is correct:

```
x1:    3    k: 2    index:   7    fractional part: 0.285714285714
x1:   15    k: 2    index:  67    fractional part: 0.268656716418
x1:   41    k: 2    index:  17    fractional part: 0.235294117647
x1:   42    k: 2    index:  59    fractional part: 0.559322033898
x1:   50    k: 2    index:  67    fractional part: 0.268656716418
x1:   67    k: 2    index: 109    fractional part: 0.238532110092
x1:   71    k: 2    index:  17    fractional part: 0.235294117647
x1: 1958    k: 2    index:  17    fractional part: 0.235294117647
x1: 1992    k: 2    index:  17    fractional part: 0.235294117647
x1: 2008    k: 2    index:  61    fractional part: 0.983606557377
x1:   99    k: 2    index: 103    fractional part: 0.650485436893
x1:    2    k: 3    index:  89    fractional part: 0.460674157303
```

Thanks.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #39 Posted by **Valentin Albillo** on 9 Apr 2008, 6:31 a.m.,*
in response to message #38 by Egan Ford

Hi, Egan:

Egan posted:

> *"can you confirm if the following is correct:"*
>
> I'll try. I can't confirm the fractional parts right now (will do tonight) but as for the indexes let's see:
>
> ```
> x1:    3    k: 2    index:   7   OK
> x1:   15    k: 2    index:  67   OK
> x1:   41    k: 2    index:  17   OK
> x1:   42    k: 2    index:  59   OK
> x1:   50    k: 2    index:  67   No, 34
> x1:   67    k: 2    index: 109   No, 51
> x1:   71    k: 2    index:  17   OK
> x1: 1958    k: 2    index:  17   OK
> x1: 1992    k: 2    index:  17   OK
> x1: 2008    k: 2    index:  61   OK
> x1:   99    k: 2    index: 103   OK
> x1:    2    k: 3    index:  89   OK
> ```
>
> It's strange that your results differ from mine in two cases about the middle of the table.
>
> I've tried my algorithm in two different implementations, one written in HP-71B's BASIC, the other written in a math software package's language, and both absolutely agree for the 200+ cases I've tried, which include values of x_1 from 2 to 15 plus 41, 42, 50, 67, 71, 1958, 1992, 2008 and assorted others, and for values of k ranging from 2 to 10.
>
> This doesn't preclude the possibility of an error on my part, of course, so please check your own implementation for the cases in disagreement (50 and 67) and if you find nothing wrong with your code e-mail me at the address given in my calc web page (don't forget to include "HP CALCS" somewhere) and we can probably sort it out. Also, if you need more test results for other values of x_1 or other values of k, let me know and I'll provide them to you ASAP.
>
> Thanks for your interest and

Best regards from V.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #40 Posted by **Eamonn** on 9 Apr 2008, 8:12 p.m.,*
in response to message #39 by Valentin Albillo

Hi Valentin, Egan,

I have implemented my solution on a PC with Python and I get the same results as Egan for all the test cases

```
x1 =    3, k = 2, n =    7, Fractional Part = 2/7
x1 =   15, k = 2, n =   67, Fractional Part = 18/67
x1 =   41, k = 2, n =   17, Fractional Part = 4/17
x1 =   42, k = 2, n =   59, Fractional Part = 33/59
x1 =   50, k = 2, n =   67, Fractional Part = 18/67
x1 =   67, k = 2, n =  109, Fractional Part = 26/109
x1 =   71, k = 2, n =   17, Fractional Part = 4/17
x1 = 1958, k = 2, n =   17, Fractional Part = 4/17
x1 = 1992, k = 2, n =   17, Fractional Part = 4/17
x1 = 2008, k = 2, n =   61, Fractional Part = 60/61
x1 =   99, k = 2, n =  103, Fractional Part = 67/103
x1 =    2, k = 3, n =   89, Fractional Part = 41/89
```

One of the things I noticed is that I needed to be careful where I applied the modulo operation. To recap, I am trying to determine if x(b) is divisible by b. I find x(n) mod b for all n < b. I use x(n) mod b to calculate x(n+1) mod b, all the way to x(b) mod b.

My first effort applied the modulo operator after calculating n*x(n) + x(n)^k. This gives (((n+1) * x(n+1)) mod b). I then used this to calculate (x(n+1) mod b) by adding multiples of b to (((n+1) * x(n+1)) mod b) until it was divisible by (n+1). I divided this by n+1 and took the result modulo b.

However, in some cases, this leads to ambiguities. For example in the case of x(1)=67 and k=3, when testing x(54) for divisibility by 54, I start off by calculating x(2) mod 54, x(3) mod 54, etc. Using the above approach, I get:

```
(2 * x(2)) mod 54 = (67 + 67*67) mod 54
                  = 20
x(2) mod 54 = 10
```

On to x(3)

```
(3 * x(3)) mod 54 = (20 + 10*10) mod 54
                  = 12
```

So far so good. Now, what is x(3) mod 54? It can actually be one of three values: 4, 22 or 40. My algorithm would find the first answer, 4 and use that in subsequent calculations. However, if I multiply everything out, I find that x(3) = 1,731,280, and x(3) mod 54 is actually 40.

I got around this by calculating (n+1) * x(n+1) from the previous values for ((n * x(n))mod b) and (x(n) mod b) and using this to calculate x(n+1) mod 54 using the same approach as above. For x(3), the calculation is:

```
3 * x(3) = 120
dividing both sides by 3 gives x(3) = 40, and x(3) mod 54 = 40.
Next, I calculate (3*x(3)) mod 54, which is 12 as before.
```

In this case, it correctly calculates x(n+1) mod 54 as 40.

Problem is, I don't know if the new approach gives the actual value for x(n+1) mod b in all cases. It gives a value that satisfies the equation, but in some cases, there may be multiple such values and I am not sure that this is the correct answer in all cases.

If anyone is interested, let me know and I can post the python code. (Any calculators out there that can run python?)

Best Regards,

Eamonn.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #41 Posted by **Egan Ford** on 10 Apr 2008, 1:27 a.m.,*
in response to message #40 by Eamonn

> Quote:
>
> I have implemented my solution on a PC with Python and I get the same results as Egan for all the test cases

This is a great case of *two wrongs don't make it right*. I think we are using the same methods and getting the same wrong answers for 2 of the cases.

> Quote:
>
> If anyone is interested, let me know and I can post the python code.

I used Perl. But will convert to 71B or 42S when satisfied with my solution. BTW, if you like Python and math you may want to check out Sage. I have a Sage server at home, but there are free ones online (https://www.sagenb.org/).

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #42 Posted by **Valentin Albillo** on 10 Apr 2008, 8:05 a.m.,*
in response to message #41 by Egan Ford

Hi, Egan:

> Just a few assorted results for various values of X_1 and K as produced by my algorithm, to help you and Eamonn check yours:
>
> | X_1 | K=2 | K=3 | K=4 | K=6 | K=8 | K=10 |
> |---|---|---|---|---|---|---|
> | 15 | 67 | 197 | 173 | 37 | 37 | 31 |
> | 41 | 17 | 79 | 17 | 83 | 23 | 23 |
> | 42 | 59 | 151 | 23 | 37 | 17 | 31 |
> | 50 | 34 | 193 | 13 | 37 | 23 | 41 |
> | 67 | 51 | 89 | 97 | 37 | 23 | 83 |
> | 71 | 17 | 79 | 17 | 13 | 46 | 89 |
> | 1958 | 17 | 158 | 13 | 13 | 149 | 41 |
> | 2008 | 61 | 229 | 167 | 13 | 37 | 31 |
>
> Keep in mind that I can't be 100% sure these are fully correct as my algorithm and/or implementations might be buggy and this is original research, never-before-seen results as far as I know, so there's no published results that I'm aware to refer to.
>
> However, though not 100%, I'm reasonably sure of their correctness as my algorithm is simple enough (for the HP-71B implementation it's a mere 5-line looping construction involving just 6 arithmetic operations inside), and I've implemented it in two very different languages and architectures, with both implementations fully agreeing in each and everyone of the 200+ assorted cases tested, which gives me a fair confidence that there's no great chance of there being an error. But it can happen, caveat emptor.

Best regards from V.

Edited: 10 Apr 2008, 8:15 a.m.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #43 Posted by **Egan Ford** on 10 Apr 2008, 10:37 a.m.,*
in response to message #42 by Valentin Albillo

Thanks.

My output (differences in bold):

x1	k=2	k=3	k=4	k=6	k=8	k=10
15	67	197	173	37	37	31
41	17	79	17	83	23	23
42	59	151	23	37	17	31
50	**67**	193	13	37	23	41
67	**109**	89	97	37	23	83
71	17	79	17	13	**67**	89
1958	17	**193**	13	**7**	**5**	**53**
2008	61	229	167	**3**	**5**	**89**

I'm 100% sure its on my end. I confirmed with arbitrary precision code that x1=2008, k=6 is not 3.

For 50,2 and 67,2 if you could tell me what modulo you used? Is it 17^2 and 2*17^2 respectively? That may help me find my bug.

Thanks.

Edited: 10 Apr 2008, 11:08 a.m.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #44 Posted by **Valentin Albillo** on 10 Apr 2008, 11:18 a.m.,*
in response to message #43 by Egan Ford

Hi again, Egan:

Egan posted:

> *"Thanks. [...] I'm 100% sure its on my end."*
>
> You're welcome. And yes, I'm afraid something's wrong in your algorithm because, just for instance, for **K = 6** and $\mathbf{X_1 = 2008}$ you're getting **3** as the index of the first non-integer element, while in fact we have:
>
> $X_1 = 2008,$
>
> $X_2 = 32775721083076740076,$ integer,
>
> and finally
>
> $X_3 = 413230412070500434091017858379218242685361747527131392695857693393246688375057739098922315422691000940697924992949576,$

which is also integer. X_4 = 124478...4462326 is integer as well, so **3** can't possibly be the correct index of the first non-integer element. Same with indexes **5** and **7** in your results, which can also be checked directly (if laboriously).

Good luck finding the (possibly very subtle) error, and

Best regards from V.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #45 Posted by **Valentin Albillo** on 10 Apr 2008, 11:34 a.m.,*
in response to message #43 by Egan Ford

Hi, Egan:

I see you edited your post while I was "in transit" composing my reply, to add the (6,2008) comment and the question, so I didn't see both edits till I clicked the submit button and the thread refreshed.

As for the question, I'm afraid I can't help. It seems my algorithm and yours are sufficiently dissimilar that your question doesn't apply to mine. I do several modular operations with different modulus in a single loop iteration, and it doesn't seem easy to discuss the details without disclosing the whole algorithm as it is implemented with so few basic arithmetic operations (just 6 per cycle).

Best regards from V.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #46 Posted by **Eamonn** on 10 Apr 2008, 1:05 p.m.,*
in response to message #43 by Egan Ford

Egan, Valentin,

Here is the output of my algorithm for the test cases above:

```
 x1    k= 2   k= 3   k= 4   k= 6   k= 8   k=10
-----------------------------------------------
  15     67    197    173     37     37     31
  41     17     79     17     83     23     23
  42     59    151     23     37     17     31
  50     67    193     13     37     23     41
  67    109     89     97     37     23     83
  71     17     79     17     13     67     89
1958     17    193     13     13     -1     41
2008     61    229    167     13     37     31
```

The -1 in the result for x1=1958, k=8 is because in one of the loops, my algorithm doesn't find a solution to xn mod b when given n*xn mod b. So it would appear that my algorithm still needs some refinement.

Of course the results I get don't fully match either of your sets of results. Here are the cases in which the three algorithms yield different results:

```
x1   k   Valentin  Egan   Eamonn
---------------------------------
  50  2     34      67     67
```

```
  67  2     51     109    109
  71  8     46      67     67
1958  3    158     193    193
1958  6     13       7     13
1958  8    149       5
1958 10     41      53     41
2008  6     13       3     13
2008  8     37       5     37
2008 10     31      89     31
```

We all get the same results for all the other cases. Based on the previous messages, I was thinking my algorithm would match Egans results in all cases - but sometimes it agrees with Valentins results. Clearly we are all doing something slightly different. I'll see if I can figure out why my routine fails in the (1958,8) case.

Best Regards,

Eamonn.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #47 Posted by **Egan Ford** on 19 Apr 2008, 7:30 a.m.,*
in response to message #43 by Egan Ford

Getting closer:

Expected:

X_1	K=2	K=3	K=4	K=6	K=8	K=10
15	67	197	173	37	37	31
41	17	79	17	83	23	23
42	59	151	23	37	17	31
50	**34**	193	13	37	23	41
67	**51**	89	97	37	23	83
71	17	79	17	13	**46**	89
1958	17	**158**	13	13	149	41
2008	61	229	167	13	37	31

Obtained:

x1	k=2	k=3	k=4	k=6	k=8	k=10
15	67	197	173	37	37	31
41	17	79	17	83	23	23
42	59	151	23	37	17	31
50	**67**	193	13	37	23	41
67	**109**	89	97	37	23	83
71	17	79	17	13	**67**	89
1958	17	**193**	13	13	149	41
2008	61	229	167	13	37	31

I found some time early this AM (jet lagged) to look at this again. Although my Perl code and Eamonn's Python code follow the same principal, Python has the benefit of arbitrary precision integers, whereas my Perl prototype does not. This explains most of the discrepancies in the lower right corner of my previous post.

The differences are now limited to those solutions with composite index numbers.

Edited: 19 Apr 2008, 8:53 a.m.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #48 Posted by **Eamonn** on 10 Apr 2008, 1:14 p.m.,*
in response to message #41 by Egan Ford

Hi Egan,

Thanks for the tip about Sage - I have not come across it before. I will investigate some more.

For my day job, I do quite a bit of numerical processing with Python, using Numpy and Scipy to simulate wireless communications systems. I used to use Matlab, which is great for scripting small stuff, but it's not an ideal language for large projects.

I have never used Perl. When I was looking around for a good general purpose scripting language, Python won out with its clean syntax and libraries for numerical computations. They both appear to be equally capable though.

Best Regards,

Eamonn.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #49 Posted by **Egan Ford** on 11 Apr 2008, 3:59 p.m.,*
in response to message #40 by Eamonn

> Quote:
>
> If anyone is interested, let me know and I can post the python code.

Please do.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #50 Posted by **Eamonn** on 11 Apr 2008, 10:07 p.m.,*
in response to message #49 by Egan Ford

Hi Egan,

Here it is. The function findit takes x1 and k as the input parameters and returns the index and the remainder. I have not had time to figure out why it fails for the (1958,8) test case since my last post. If you wouldn't mind posting your perl code for comparison that would be great.

A couple of tips for anyone trying to follow the code that hasn't seen python before:

- Python doesn't use { and } as in c to group lines of code into a block. Instead, it treats lines of code at the same indentation as a single block of code.

- % is the python modulo operator (eg. 17 % 5 = 2)

- ** is the python power operator (eg. 2 ** 5 = 32)

- # is the python comment indicator

Best Regards,

Eamonn

```
def findit(x1, k):
    a = 1
    done = False


    while done == False:
        d   = 0
        xn  = x1
        nxn = xn
        a += 1
        for n in range(1, a):   # c equivalent is: for ( n=1 ; n<(a-1) ; n++ )
            np1xnp1 = nxn + xn ** k
            d = 0
            if a != n+1:        # search below will fail if a == n+1
                while (d*a + np1xnp1) % (n+1) != 0 and done == False:
                    d+=1
                    if d > n+1: # Failed to find a solution to the modulo problem
                        done = True
                        a = -1


                xn = ( ( d * a + np1xnp1 ) / (n+1) ) % a
            np1xnp1 = np1xnp1 % a
            nxn = np1xnp1
        if nxn != 0:
            done = True
    return (a, nxn)
```

Edited to fix small cut/paste error in original posted code

Edited: 12 Apr 2008, 12:38 a.m.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #51 Posted by **Eamonn** on 14 Apr 2008, 1:41 a.m.,*
in response to message #50 by Eamonn

Egan, Valentin,

My intention was to port the above algorithm or a slightly modified version to the HP-71B. After examining some more cases however, it is clear that the problem of multiple solutions to xn mod a, when (n * xn) mod a is known, is the root of the problem with this approach.

I made an attempt at trying to decide between the possible solutions by making use of the previously calculated and stored solution to xn mod (a-1). However there are cases where this approach also fails to find a unique solution for xn mod a.

This is certainly an interesting problem to work on. I now know a lot more about modular arithmetic that before, having now come across the linear congruence theorem, the extended Euclidean algorithm and the Chinese remainder theorem. Still not quite there yet with a solution.

From your previous posts Valentin, it appears that you are using perhaps a different approach. I look forward to when you publish your solution, or to Egan's or anyone else's solution also.

Best regards,

Eamonn.

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #52 Posted by **PeterP** on 14 Apr 2008, 11:04 p.m.,*
in response to message #51 by Eamonn

Eamonn,

Sorry to bug you, but would you possibly be so kind and post (or email) me the key pieces you found to learn about the items you mentioned about modular arithmetic? I never learned it in school yet found it very interesting.

Thanks for your and Egan's ongoing posting on this problem, I'm a silent follower...

Cheers

Peter

Re: SSMC20 Challenge 5. (answers w/ question)

*Message #53 Posted by **Eamonn** on 15 Apr 2008, 11:46 p.m.,*
in response to message #52 by PeterP

Hi Peter,

No problem - I found most of the information searching for "modulo arithmetic". The most straightforward explanations that I found are on Wikipedia at the links below. I hope you find them useful.

By the way, I was reading through your solution to Problem #4. I see that you have some modular arithmetic in there, so you seem to know a thing or two about it :-) I'll have to study your solution a bit more to fully understand it, but it looks pretty impressive.

Best Regards,

Eamonn

Modular Arithmetic

Linear Congruence Theorem

Extended Euclidean Algorithm

Chinese Remainder Theorem

Challenge 3

*Message #54 Posted by **PeterP** on 10 Apr 2008, 3:40 a.m.,*
in response to message #1 by Valentin Albillo

Hmm, I think I am starting to see why you love the 71b. After I was able to wrap my head around the problem, it was relatively straight forward to come up with a basic type code but implementing this in RPN proved cumbersome. So, yet again, we have a very looong code. However, I do admit that I have not spent any time on improving the actual coding as I think I spent too much time in thinking on how to cut down execution time. Again, something one would need not worry too much with the 71 or 42. Alas, I am still not very apt in using the 71 (and have never used the 42 actually although I was lucky and procured a pristine exemplar a year back or so).

Ok, here is my approach, slow and long-winded as it is:

1. Peter's first statement tells us that his product must have at least 3 prime factors (it was good I looked at #5 for a while...) which means that there are at least 2 numbers that Ms Germain could have thought off. if we assume that the product has 3 prime factors p1, p2, p3, the numbers could be (p1,p2*p3) or (p1*p2,p3)
2. Susan's statement is where the actual meat was hidden at least for me. Her statement 'I knew that already' tells us that her number can not be represented as a sum of 2 primes. This reminded me on a day I brought down the complete network of a large company I was working for as a summer-intern and had been given programming access to the mainframe (sexy RPL/II). And used it to test something that I believe is called Goldbach's assumption (at least in German) - every even number can be represented as a sum of 2 primes.

So from this little exchange we know the following (at least I think so...) about the set of sums Susan might have been given, lets call it S

1. It has no even numbers
2. Any member of S - 2 can not be a prime number (as 2 is a prime number as well, the product would be a product of two prime numbers and hence unique)
3. The smallest possible number is 11

So we can write a little program that generates the whole set S and stores it in a sequence of registers for convenience. The code to do so and the resulting set S is below

```
S={11,17,23,27,29,35,37,41,47,51,53,57,59,65,67,71,77,79,83,87,89,93,95,97}

LBL 'MSL
SIZE?
50
X>Y?
PSIZE
10
STO 01
Enter
Enter
*
*
STO 05
2
STO 06
11.09902
STO 00
LBL 00
  RCL 00
  INT
  RCL 06 (:=2)
  -
  PRIME?    (I know cheating,
    GTO 14  This is from Angel Martins wonderful
  XEQ 'IP   Sandbox ROM. Forgive me Valentin...)
  LBL 14
  ISG 00
GTO 00
GTO 'FIND
```

```
LBL 'IP
  ISG 01
  NOP
  RCL 00
  INT
  View X
  STO IND 01
  RCL 06 (:=2)
  /
  INT
  RCL 05 (:=1000)
  /
  ST+ IND 01
RTN
```

From here we proceed somewhat with brute force yet try to cut off unnecessary tests and loops (this is what proved really hard to implement on the 41. Maybe if I have time I try to write a solution for the 71 which should be much shorter...)

1. Pick a member s_i from the set S
2. Generate all possible pairs (a_t,b_t) for which $a+b = s_i$
3. Calculate the respective product $p_t = a_t*b_t$ for all pairs (a_t,b_t) for all s_i element of S

Now we need the remaining statements from Peter and Susan. Peter claims in his next statement that he now knows the numbers. This means that there is a value p_t that only appears once. But which one? there are still many values p_t that appear only once. Luckily, after his statement, Susan now also claims that she knows the numbers. This means that p_t is unique and is the only unique p_t for that s_i.

The following code implements just that, using a few short-cuts, namely

1. When testing to see if a p_t of a given s_i is unique, we loop over all s_k and see if any of their respective (a_t,b_t) can generate the same p_t. To do so, we don't have to loop over all pairs as we can simply solve the quadratic equation $(s_k-x)*x = p_t$. And actually, we do not have to solve the full equation but just check if the $s_k^2-4*p_t$ is a perfect square. I start to see the next challenge rear its ugly head...
2. Further more we can stop looking for possible matches once $(s_k-2)*2 > p_t$ as the products are monotonic increasing.
3. Lastly we can abandon to check the possible pairs (a_t,b_t) for a particular s_i if we have found more than one product p_t that is unique for this s_i
4. Lastly, I keep track off how many duplicate products each s_t produces and if one has only duplicates, I actually remove it from the test. I thought originally that most p_t would be duplicates but it turns out they are not so this part of the code could be removed without a great loss of speed or any at all.

As it turns out, there is indeed only one solution which my trusty HP41 spits out after about 50sec or so. It then continues to rattle along for another 18 minutes or so to finally declare that this is the only solution. Horray.

Ah yes, I believe the solution to be (4,13).

And for those who really want to see it, here is the full code. Typing it in here I realized that I really need to clean this up. This is ugly...

```
LBL FIND
Tone 5
RCL 01
RCL 05
/
11
+
STO 00
STO 08
LBL 01 'loop over all sums s
```

```
CLX
STO 02 'this is my counter of unique products p
RCL IND 00 'this holds the sums s
INT
View X
RCL 06 (:=2)
/
INT
RCL 05 (:=1000)
/
RCL 06
+
STO 03
LBL 03 'loop over all pairs (a,b)
  RCL IND 00
  FRC
  x=0?
    GTO 91 'This sum has no free products left
  LastX
  INT
  RCL 03
  INT
  -
  LastX
  *
  4
  *
  STO 07    '4*p for the sqrt(s^2-4t)?=?integer
  SF 00 'lets assume it is unique
  RCL 08 'holds the counter to loop over all possible sums
  STO 04
  LBL 04 'loop over all possible sum
    RCL IND 04
    INT
    RCL IND 04
    INT
    x=y? 'don't look here if this is the same sum we came from
      GTO 15
    x^2
    RCL 07 '=4*p
    -
    x<0?
      GTO 13 'clearly no solution...
    SQRT
    FRC
    x=0?
      XEQ 'NU => Not unique
    LBL 13
    RCL 04
    RCL 00
    x<=y?
      XEQ 'TG =>Test if further s will definitely be too big and have no matches
    LBL 15
    ISG 04
  GTO 04
  LBL 93
  FS?C 00
    XEQ 'IU => This is a unique pt
  ISG 03
GTO 03
```

```
  LBL 91
  RCL 02
  x=1?
    XEQ'FS ======> YES! we found a solution! only one unique pt for this sum
  ISG 00 'but keep checking to make sure it is unique
GTO 01
TONE 0
'DONE
AVIEW
STOP
GTO'MSL


LBL TG 'Check if we need to keep looking for greater sums
  RCL IND 04
  INT
  RCL 06 (:=2)
  -
  LastX
  *
  RCL 07 (:=pt)
  4
  /
  x>y?
    RTN
  CLX
  STO 04 'this kills the loop at the next ISG 04
RTN


LBL'NU  'Not Unique pt, subtract available pairs from s to reduce
  RCL 05
  1/x
  ST- IND 04
  CF 00
  CLX
  STO 04
RTN


LBL'IU   'This is a unique solution. Remember it!
  Tone 9
  RCL IND 00
  INT
  RCL 03
  INT
  -
  LastX
  RCL 05
  /
  +
  STO 49
  RCL 02
  INCX
  STO 02
  x=1?
    RTN
  CLX 'too bad, we now have more than one solution for this s...
  STO 03
RTN
```

```
LBL'FS====> Proudly show the solution we found
  BEEP
  View 49
STOP
RTN
```

Edited: 10 Apr 2008, 3:43 a.m.

Re: Challenge 3

*Message #55 Posted by **J-F Garnier** on 10 Apr 2008, 7:18 a.m.,*
in response to message #54 by PeterP

Hi Peter,

I'm still working on my program (not yet running), but I think your result is correct.

Let's try to verify how Susan and Peter were thinking:

```
P=52, so Peter can't determine if the numbers are (2,26) or (4,13).


S=17, so Susan can't determine if the numbers are:
  (2,15), (3,14), (4,13), (5,12), (6,11), (7,10) or (8,9)
and she is sure that in any case Peter can't determine the numbers because none of her candidate number pairs is made of 2 prime numbers.


As Peter now knows that, he can think like this:
His (2,26), (4,13) candidate number pair makes the following possible sum: 28 and 17.
  If the sum would be 28, would Susan be able to be sure that Peter can't determine the numbers? No, because with a sum of 28 the Susan's candidate number would be:
  (2,26), (3,25), (4,24), (5,23), (6,22), (7,21) ...
and (5,23) is made of 2 prime numbers.
So now, Peter KNOWS that the sum is 17, and thus knows the numbers are 4 and 13.


Is now Susan able to find the numbers?
Her candidate numbers are  (2,15), (3,14), (4,13), (5,12), (6,11), (7,10) or (8,9), so the possible products are: 30, 42, 52, 60, 66, 70 and 72.
If Peter was able to find the number in only one case, then it would tell Susan the solution.
Would Peter have been able to find the numbers in another case than the correct P=52?


If the product would be 30, the possible candidates of Peter would have been (2,15) and (3,10), (6,5) and Peter sum's candidate would be 17, 13, 11.
  With all these odd sums, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would not have find the numbers.


If the product would be 42, the possible candidates of Peter would have been (2,21) and (3,14), (6,7) and Peter sum's candidate would be 23, 17, 13.
  With all these odd sums, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would not have find the numbers.


If the product would be 52, the possible candidates of Peter would have been (2,26) and (4,13) and Peter sum's candidate would be 28, 17.
  With 28, Susan would not be sure because 28  may be = 5+23 (prime numbers),
  With 17, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would have find the numbers (4,13) [that he actually did].
```

```
If the product would be 60, the possible candidates of Peter would have been (2,30) and (3,20), (4,15), (5,12), (6,10) and Peter sum's candidate would be 32, 23, 19, 17 and 16.
  With 32, Susan would not be sure because 32 may be = 3+29 (prime numbers),
  With 16, Susan would not be sure because 16 may be = 3+13 (prime numbers).
  With the other odd sums, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would not have find the numbers.


If the product would be 66, the possible candidates of Peter would have been (3,22) and (6,11) and Peter sum's candidate would be 25, 17.
  With it, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would have find the numbers.


If the product would be 70, the possible candidates of Peter would have been (5,14) and (7,10) and Peter sum's candidate would be 19, 17.
  With it, Susan would be sure because no odd number is the sum of 2 prime numbers,
So in this case, Peter would have find the numbers.


So there is only one case where Susan can find the numbers knowing that Peter did it.
```

The numbers are 4 and 13, if I'm correct.

Valentin, can you confirm if this is correct?

Edited: 10 Apr 2008, 7:23 a.m.

Re: Challenge 3

*Message #56 Posted by **Valentin Albillo** on 10 Apr 2008, 7:28 a.m.,*
in response to message #55 by J-F Garnier

Hi, J-F:

Jean-François posted:

> *"The numbers are 4 and 13, if I'm correct. Valentin, can you confirm if this is correct ?"*
>
> Yes I can and yes, they are correct.
>
> Congratulations both to PeterP for his insight and clever RPN program and to you for your accurate logic and detailed explanation.
>
> See to it that an HP-71B program is created to find the solution, nothing less is to be expected from Emu71's creator. :-)

Best regards from V.

Re: Challenge 3

*Message #57 Posted by **PeterP** on 10 Apr 2008, 12:05 p.m.,*
in response to message #55 by J-F Garnier

wow, this is a brilliantly clear explanation, J-F. I wish I had had you as a teacher J-F, I'm very impressed.

Attention, potential spoiler - having finished the challenge I did allow myself to google for Mrs Germain and found this link to learn that she is actually quite famous and has a very moving and impressive live. And her first name is Sophie...

Cheers

Peter

Re: Challenge 3

*Message #58 Posted by **J-F Garnier** on 10 Apr 2008, 3:21 p.m.,*
in response to message #55 by J-F Garnier

Below is my program for the HP-71B. I used an idea from Peter, i.e. a even number is always the sum of 2 prime numbers, this avoids to explicitly test for prime numbers.

I found the same solution than Peter: (4,13) but the surprise is that my program finds three other possible solutions. I checked the (13,16) pair by hand in a similar way than I checked the (4,13) pair above, and it seems OK. Can somebody confirm it with another method, or did I something wrong?

J-F

```
10 ! --- S&SMC 20 ---
20 ! JFG tentative solution
30 FOR S1=7 TO 100 STEP 2 ! for all plausible sums
40 C2=0
50 ! consider all possible pairs (A1,B1) for the given sum S1
60 FOR A1=2 TO S1/2
70 B1=S1-A1
80 P1=A1*B1 ! tentative product
90 ! consider all the pairs for the tentative product P1
100 C=0
110 FOR A2=MAX(2,INT(P1/99)) TO SQRT(P1)
120 IF MOD(P1,A2)#0 THEN 180
130 B2=P1/A2
140 S2=A2+B2
150 IF MOD(S2,2)=0 THEN 180
160 C=C+1 ! count the odd sums
170 IF C=1 THEN A8=A2 @ B8=B2 ! store possible solution
180 NEXT A2
190 IF C=1 THEN A9=A8 @ B9=B8 @ C2=C2+1 ! one possible solution
200 NEXT A1
210 IF C2=1 THEN DISP A9,B9 ! this is a solution
220 NEXT S1
>RUN
 4                    13
 13                   16
 4                    37
 16                   37
```

Edited: 10 Apr 2008, 3:35 p.m.

Re: Challenge 3

*Message #59 Posted by **Valentin Albillo** on 10 Apr 2008, 7:46 p.m.,*
in response to message #58 by J-F Garnier

Hi, J-F:

Jean-François wrote:

> *"I checked the (13,16) pair [...] and it seems OK. Can somebody confirm it with another method, or did I something wrong ?"*
>
> Sorry, J-F but (13,16) isn't another solution, your program is nice but slightly simpler than needed here.
>
> The pair (13,16) has sum 29. With that sum the first three dialogs would be fulfilled but not the fourth, i.e, Peter, given the product of 13*16 = 208 would be able to deduce the correct pair after Susan's first comment. But the fourth dialog cannot be fulfilled: Susan has no way to know if the correct sum, 29, came as 2+27 or 13+16. Think about it.

Thanks for your interest and

Best regards from V.

Re: Challenge 3 - corrected

*Message #60 Posted by **J-F Garnier** on 11 Apr 2008, 8:15 a.m.,*
in response to message #59 by Valentin Albillo

OK, I see my mistake, my assumption "*no odd number is the sum of 2 prime numbers*" is obviously false...

Here is my corrected program, a bit less simple as I had to explicitly test for some prime numbers.

J-F

```
 10 ! --- S&SMC 20 ---
 20 ! JFG corrected solution
 30 FOR S1=5 TO 195 STEP 2 ! for all plausible sums
 35   IF FNP(S1-2) THEN 220
 40   C2=0
 50   ! consider all possible pairs (A1,B1) for the given sum S1
 60   FOR A1=2 TO S1/2
 70     B1=S1-A1
 80     P1=A1*B1 ! tentative product
 90     ! consider all the pairs for the tentative product P1
100     C=0
110     FOR A2=MAX(2,INT(P1/99)) TO SQRT(P1)
120       IF MOD(P1,A2)#0 THEN 180
130       B2=P1/A2
135       IF B2>99 THEN 180
140       S2=A2+B2
150       IF MOD(S2,2)=0 THEN 180
155       IF FNP(S2-2) THEN 180
160       C=C+1 ! count the items not a sum of 2 prime numbers
170       A8=A2 @ B8=B2 ! store possible solution
180     NEXT A2
190     IF C=1 THEN A9=A8 @ B9=B8 @ C2=C2+1 ! one possible solution
200   NEXT A1
210   IF C2=1 THEN DISP A9,B9 ! this is a solution
220 NEXT S1
230 END
240 DEF FNP(X)  ! test if X is prime (X odd, X<200)
250   FNP=0
```

```
260   IF X=3 OR X=5 OR X=7 OR X=11 OR X=13 THEN FNP=1
270   IF MOD(X,3) AND MOD(X,5) AND MOD(X,7) AND MOD(X,11) AND MOD(X,13) THEN FNP=1
280 END DEF
>RUN
 4                    13
```

Edited: 12 Apr 2008, 10:43 a.m.

Challenge 4

*Message #61 Posted by **PeterP** on 12 Apr 2008, 9:52 p.m.,*
in response to message #1 by Valentin Albillo

I approached the 4th challenge first with paper and pencil. Soon I was back to prime-factorizations and developed a matrix representation scheme that proved quite useful.

- Lets call our input numbers n^i with i=1,2,...NMax. NMax := 5 for the stated problems and we'll generaly use super-scripts for our input-numbers
- Let $P = (p_1,p_2,...,p_{np})$ be a vector which holds all prime-factors that appear at least in one of the input numbers n^i
- So each $n^i = \text{product}(p_1^{k_1^i},p_2^{k_2^i},...,p_{np}^{k_{np}^i})$
- We can now arrange the prime-factorization of our input numbers in a neat matrix PFM (PrimeFactorizationMatrix) with np rows and NMax columns.
- with pfm(row,column) = k_{row}^{column}

so for example for the set (50,1958,2008) we would have

```
P = (2,5,11,89,251)
      |1  1  3|
      |2  0  0|
PFM = |0  1  0|
      |0  1  0|
      |0  0  1|
```

as

$50 = 2^1*5^2*11^0*89^0*251^0$
$1958 = 2^1*5^0*11^1*89^1*251^0$
$2008 = 2^3*5^0*11^0*89^0*251^1$

Next I decided to respresent the constant we are asked to find $C(n^1,n^2,...,n^{NMax})$ in its prime factorization form using the elements of P. This means we have to find a seperate c_i for each row of PFM. Sounds like it has gotten worse, now we are looking for np constants...

To generate perfect powers, there needs to be a value pp^i for each number n^i for which k_j^i MOD $pp^i = 0$ for all j=1,2,...,np

The following algorithm does most likely not find the smallest constat $C = \text{product}(p_1^{c_1},p_2^{c_2},...,p_{np}^{c_{np}})$ as I have arbitrarily assigned $pp_1 = 2, pp_2 = 3, pp_3 = 5$, etc. I believe this to be not a horrible choice but I'm sure one can find smaller C in certain cases. Basically, depending on the relationship between the k_j^i there is an optimal set of pp_i but I could not figure out an easy way to find that set.

From here we can build a relatively fast and straight forward code which, however, I implemented in the 71b (learning a lesson from the last challenge).

1. Get the number of input numbers N1 and the input numbers N()
2. Store the pp in R() with $R_1=2, R_2=3$etc

3. Calculate the prime factorization of each number, storing the set of appearing prime-factors in P() and the PFM matrix in P1()
4. Loop over all rows j
5. Calculate the first possible integer T = [R_{N1} - k_1^{N1} Mod R_{N1}] Mod R_{N1}. I'm sure there is a prettier calculation but this one works at least...
6. Check if the sum of all the ($T+k_j^i$) Mod R_j = 0
7. If not, increase T by R_{N1}. This is similar to the Sieve of Eratosthenes
8. Next row j

Two comments to the code below. First, I am very inexperienced with using the 71b so please forgive the cluncky code. Second, I used the math-rom and the JPC rom which made the code easier to write, hoping that the kind author of the challenge will overlook this.

The code produces the following answers

```
C(41,67)          = 41^3 * 67^2
C(1776,2008)      = 3^3 * 37^3 * 251^2
C(2007,2008)      = 223^3 * 251^2
C(50,1958,2008)   = 2^17 * 11^20 * 89^20 * 251^24
C(2,3,4)          = 2^3 * 3^20
C(3,4,5)          = 3^15 * 2^10 * 5^24
C(2,3,4,5)        = 2^63 * 3^140 * 5^90
C(2,3,4,5,6)      = 2^483 * 3^560 * 5^1980
C(10,11,12,15,16) = 2^2163 * 5^825 * 11^770 * 3^594
C(71,42,15)       = 71^15 * 2^20 * 3^14 * 7^20 * 5^24
```

Listing (mod typos)

```
10 Option Base 1 @ Destroy All
20 Input "Nr of Integers: ";N1
30 Dim R(6) @ R(1)=2 @ R(2)=3 @ R(3)=5 @ R(4)=7 @ R(5)=11 @ R(6)=13
40 Dim N(N1) @ Dim P(20) @ Dim P1(20,N1) @ Dim R(N1)
50 Mat Input N @ k = 0
55 REM Calculate all primfactors P and matrix PFM (:=P1)
60 For i = 1 To N1
70     X = PRIM(N(i)) @ If X = 0 Then X = N(i)
80     While N(i) <> 1
90         k = k+1 @  P(k) = X
100        For j = 1 To N1 @ P1(k,j) = FNC(N(j),X) @ N(j) = N(j)/X^P1(k,j) @ Next j
110        X = PRIM(N(i)) @ If X = 0 Then X = N(i)
120    End While
130 Next i @ Dim P1(K,N1) @ Dim P(K)
135 REM Calculate constant, prime-factor by prime-factor
140 For L = 1 to K
150     X = Mod(P1(L,N1),R(N1))
160     T = Mod(R(N1)-X,R(N1))
170     S = 0 @ For j = 1 to N1 @ S = S + Mod(T+P1(L,j),R(j)) @ Next j
180     While S <> 0
190         T = T + R(N1)
200         S = 0 @ For j = 1 to N1 @ S = S + Mod(T+P1(L,j),R(j)) @ Next j
210     End While
220     Disp P(L);"^";T;" * "
230 Next L
240 Disp "1. Done!" @ Beep
500 Def FNC(N,F) @ C=0 @ While FP(N/F) = 0 @ C=C+1 @ N=N/F @ End While @ FNC=C @ End Def
```

Edited: 16 Apr 2008, 10:22 a.m. after one or more responses were posted

Re: Challenge 4

*Message #62 Posted by **J-F Garnier** on 14 Apr 2008, 7:28 a.m.,*
in response to message #61 by PeterP

Hi Peter,

I can't make your program running, probably some typos...

J-F

Re: Challenge 4

*Message #63 Posted by **PeterP** on 14 Apr 2008, 9:44 a.m.,*
in response to message #62 by J-F Garnier

J-F,

Thanks, yes there was a typo in line 100 (* instead of @). It should work now but I can't check (in the office and don't have the calc with me or your simulator...)

Thanks for letting me know!!

Cheers

Peter

Re: Challenge 4

*Message #64 Posted by **Steve Perkins** on 14 Apr 2008, 1:14 p.m.,*
in response to message #62 by J-F Garnier

Is there an "easy" or at least straight-forward way to cut and paste code like this into the simulator? I'd like to try it myself, but don't have time to retype it just now.

Thanks for the wonderful EMU71 by the way, and any help you (or others) can give!

Re: Challenge 4

*Message #65 Posted by **Valentin Albillo** on 14 Apr 2008, 2:04 p.m.,*
in response to message #64 by Steve Perkins

Hi, Steve:

Steve posted:

"Is there an "easy" or at least straight-forward way to cut and paste code like this into the simulator?"

> Well, yes ... as "easy" as copying the text listing from the web page and pasting it into Emu71 in a newly created, empty program file. Or are you saying you don't know how to copy and paste text between a Windows GUI application and a console-style one ?

Also, there's some controversy on whether to call this kind software "emulator" or "simulator". My own criterium is to call it "emulator" when it's actually using the original ROM assembler code which gets executed by an emulated CPU (Saturn for the HP-71B), and "simulator" otherwise (i.e., doesn't use the original ROMs nor does it execute original assembler code).

Using this criterium, Emu71 is an "emulator" for me, not a "simulator". And a pretty good one at that.

Best regards from V.

Re: Challenge 4

*Message #66 Posted by **Steve Perkins** on 14 Apr 2008, 8:13 p.m.,*
in response to message #65 by Valentin Albillo

You're right, of course. I should have used the term "emulator" for the excellent EMU71.

As for the copying and pasting, I should have explained my problem. I assumed the same thing afflicted everyone. When I try to paste a copied line, most of the text gets missed. Only a few characters get entered. It seems the emulator is looking for keystrokes and missing some when the come to quickly for the original ROM code to copy with perhaps?

This is in a Windows XP cmd.exe window. Does anyone else have this problem? Or does it work better in another environment?

For instance, trying to paste (from the above code) line 240, instead of

```
240 Disp "1. Done!" @ Beep
```

I get

```
24d1N"E
```

Re: Challenge 4

*Message #67 Posted by **Valentin Albillo** on 15 Apr 2008, 2:42 a.m.,*
in response to message #66 by Steve Perkins

Hi again, Steve:

The problem you describe was indeed the case with earlier versions of Emu71 but Jean-François did address the problem long time ago and newer versions do accept pasted text correctly, missing no characters (with a few exceptions).

Also, it reverts the capitalization of pasted text so that uppercase letters are pasted as lowercase and vice versa. This is transparent as far as keywords are concerned, because program listings are always generated with capitalized keywords regardless of how they were entered, but text strings will indeed change capitalization and may need manual correction. This was the case in the latest version I have, perhaps the newest one has this point straightened out.

I suggest you visit J-F's Emu71 site and download the latest version. It should correct your problem at once.

The exception has to do with control characters: some sequences involving the exponentiation operator ("^") get interpreted as control characters instead and have the operator missing. But this is rare enough and easily detected and corrected, if it's still a problem with the newest version.

Best regards from V.

Edited to correct a minor typo and further explain about capitalization reversal

Edited: 15 Apr 2008, 7:26 a.m.

Re: Challenge 4

*Message #68 Posted by **J-F Garnier** on 15 Apr 2008, 6:43 a.m.,*
in response to message #66 by Steve Perkins

Hi Steve,

Valentin explained all (Thanks, Valentin!)

Please have a look here and let me know if you still have problems.

J-F

Re: Challenge 4

*Message #69 Posted by **Steve Perkins** on 15 Apr 2008, 12:34 p.m.,*
in response to message #68 by J-F Garnier

Thanks, I'll install the latest right away.

I feel silly that this wasn't an obvious thing for me to try.

Update: works as advertised! Thanks for the advice, and especially for the fantastic software!

Re: Challenge 4

*Message #70 Posted by **Steve Perkins** on 15 Apr 2008, 2:33 p.m.,*
in response to message #61 by PeterP

Nicely done Peter!

I did have to make a correction to line 220. It seemed you meant to display entries from the P() array. And a small improvement:

```
220     If T>0 Then Disp P(L);"^";T;" * "
```

Other than that, I double checked most of the outputs and it ran as you specified. I'm getting more fond of the math and JPC roms too! I wish I had some good documentation for the math rom, since I finally got one with an HP-71B I acquired.

Very nicely analyzed and programmed.

Edited: 15 Apr 2008, 2:55 p.m.

Re: Challenge 4

*Message #71 Posted by **PeterP** on 15 Apr 2008, 4:57 p.m.,*
in response to message #70 by Steve Perkins

Steve, thanks for your kind words, improvement and identification of Typo! I was too elated & spent at that point to improve :-)

As for the math rom documentation, you can find it on the fantastic DVD that Dave hicks sells here on the museum (aside from all sorts of other goodies). Check out this link

And for the JPC rom, J-F has put together a great resource including manuals. Check out this link

Cheers

Peter

Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #72 Posted by **Valentin Albillo** on 15 Apr 2008, 7:52 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

> Within the next day or two I'll post my original solutions to the current S&SMC#20 so if you want to post yours before I do, this is your last chance.
>
> It depends of my workload and free time availability but I'd say you still have some 24 to 72 hours before the deadline. As requested, I've extended it past the one-week usual length to allow Egan, Eamonn, and all others to refine or correct their solutions, but it seems all's said and done by now.
>
> Thanks to all for your interest and willingness to share and

Best regards from V.

Re: Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #73 Posted by **Egan Ford** on 15 Apr 2008, 5:32 p.m.,*
in response to message #72 by Valentin Albillo

Hola Valentin,

Sadly, I've been swamped with work and travel for the last 5 days with no outlook of enough downtime to revisit #5 or #4. (Not far from you actually, in Sevilla now.) That said I have not given up, but fear I will run out of time.

Re: Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #74 Posted by **Valentin Albillo** on 21 Apr 2008, 11:23 a.m.,*
in response to message #73 by Egan Ford

Hi, Egan:

> I've posted my original solutions and comments for all problems except the last three, which include the ones you've been working on.
>
> Within 24 hours I'll post the remaining solutions, so I sincerely hope you'll be able to make it. It's already some 21 days since I posted the challenge and the start of the thread risks disappearing on its way towards the Archives, which would render it incomplete ... :-)

Best regards from V.

Re: Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #75 Posted by **Egan Ford** on 21 Apr 2008, 5:43 p.m.,*
in response to message #74 by Valentin Albillo

Thanks for the extra time. Unless an *Ah-ha!* strikes before I sleep I am afraid all I have is a solution that will correctly find the index only if it is prime. The composite indexes still elude me.

Incomplete 71B entry (bold are wrong):

```
K= 2  X1=  15 INDEX= 67 REM= .268656716418
K= 2  X1=  41 INDEX= 17 REM= .235294117647
K= 2  X1=  42 INDEX= 59 REM= .559322033898
K= 2  X1=  50 INDEX= 67 REM= .268656716418
K= 2  X1=  67 INDEX=109 REM= .238532110092
K= 2  X1=  71 INDEX= 17 REM= .235294117647
K= 2  X1=1958 INDEX= 17 REM= .235294117647
K= 2  X1=1992 INDEX= 17 REM= .235294117647
K= 2  X1=2008 INDEX= 61 REM= .983606557377
K= 2  X1=  99 INDEX=103 REM= .650485436893
K= 3  X1=   2 INDEX= 89 REM= .460674157303
```

Code:

```
10 DESTROY ALL @ OPTION BASE 0
20 ! INPUT "k=";K @ INPUT "x1=";X1 @ CALL DOIT(K,X1,I,R)
30 FOR J=1 TO 11 @ READ K @ READ X1 @ CALL DOIT(K,X1,I,R)
40 DISP "K=";K;" X1=";
50 DISP USING "4D,7A,3D,5A,D.12D";X1," INDEX=",I," REM=",R/I
60 NEXT J
70 END
80 DATA 2,15,2,41,2,42,2,50,2,67,2,71,2,1958,2,1992,2,2008,2,99,3,2
90 SUB DOIT(K,X1,I,R)
100 DIM P(100) @ P(0)=2 @ L=1 @ Q=2
110 'START': X=X1 @ CALL NPRIME(Q,P(),L)
120 FOR N=1 TO Q
130 X=MOD(X,Q) @ T=X
140 FOR J=1 TO K-2 @ T=T*X @ T=MOD(T,Q) @ NEXT J
150 Y=MOD(X*(T+N),Q)
160 IF Y=0 THEN GOTO 'START'
170 FOR Z=1 TO Q
180 IF MOD(Z*(N+1),Q)=Y THEN X=Z @ GOTO 210
190 NEXT Z
200 I=N+1 @ R=Y @ GOTO 220
210 NEXT N
220 END SUB
230 SUB NPRIME(Q,P(),L)
240 J=P(L-1)
250 J=J+1
260 FOR I=0 TO L-1
270 IF MOD(J,P(I))=0 THEN 250
280 NEXT I
290 Q=J @ L=L+1 @ P(L-1)=Q
300 END SUB
```

Edited: 21 Apr 2008, 6:33 p.m.

Re: Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #76 Posted by **Eamonn** on 22 Apr 2008, 2:18 a.m.,*

in response to message #75 by Egan Ford

Nice Work Egan.

I am in more or less the same boat as you - I have an algorithm that works for the cases where the index is prime. Kudos to you though for submitting an actual solution for a programmable calc.

Having spent quite a bit of time on this problem over the past few weeks, I am really looking forward to seeing Valentin's solution. I am pretty sure that I will learn a lot from it, as usual.

Best regards,

Eamonn.

Re: Last chance (Re: Short & Sweet Math Challenge #20: April 1st, 2008 Spring Special)

*Message #77 Posted by **Egan Ford** on 22 Apr 2008, 9:45 a.m.,*
in response to message #76 by Eamonn

Thanks. Here's more stuff. If you replace lines 10-80 with:

```
5 DESTROY ALL @ OPTION BASE 0 @ DIM X(6) @ DIM Y(8) @ DIM Z(6)
10 DISP "  x1";
15 FOR J=1 TO 6 @ READ X(J) @ IF X(J)<10 THEN DISP " ";
20 DISP "  k=";STR$(X(J)); @ NEXT J
25 DISP @ FOR J=1 TO 40 @ DISP "-"; @ NEXT J @ DISP
30 FOR J=1 TO 8 @ READ Y(J) @ NEXT J
35 T=TIME
40 FOR J=1 TO 8 @ FOR L=1 TO 6 @ CALL DOIT(X(L),Y(J),I,R)
45 Z(L)=I @ NEXT L
50 DISP USING "4D,6D,6D,6D,6D,6D,6D";Y(J),Z(1),Z(2),Z(3),Z(4),Z(5),Z(6)
55 NEXT J
60 DISP "TIME:";TIME-T
65 END
70 DATA 2,3,4,6,8,10
75 DATA 15,41,42,50,67,71,1958,2008
```

You can generate the following table (errors in bold):

```
  x1   k=2   k=3   k=4   k=6   k=8  k=10
--------------------------------------
  15    67   197   173    37    37    31
  41    17    79    17    83    23    23
  42    59   151    23    37    17    31
  50    67   193    13    37    23    41
  67   109    89    97    37    23    83
  71    17    79    17    13    67    89
1958    17   193    13    13   149    41
2008    61   229   167    13    37    31
TIME: 122.06
```

2 minutes with EMU71. I didn't have time to test a physical 71B. I am guessing that it would have taken hours.

If any are interested I have qbasic versions as well. The above code runs in less than a second compiled with FreeBASIC.

Edited: 22 Apr 2008, 9:52 a.m.

My Original Solutions and Comments - Part 1 (Re: S&SMC#20)

*Message #78 Posted by **Valentin Albillo** on 21 Apr 2008, 10:19 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

Thanks a lot for your very interesting solutions to the tasks featured in this last ***S&SMC#20***, which accordingly with its *'series finale'* nature were a little more tricky than usual. Nevertheless you did cope pretty well and most of them were solved very quickly and efficiently.

I've been marking time to allow **Egan Ford** an other kind contributors to complete their solutions to the most difficult problems but at long last, these are my original solutions and assorted comments:

The three-pronged pre-Challenge training:

---

1 - Simplest

"Given an arbitrary 4x4 real matrix, write a program to compute and output the sum of its four eigenvalues, real and/or complex".

This was nailed down by **Nenad** almost instantly. My original solution is this ultra-short, *3-step* RPN program valid for *any* RPN model:

```
01  +
02  +
03  +
```

where you must input the four elements from the main diagonal (all other elements are irrelevant) into the RPN stack, then run the program (or press **[+]** three times if your calculator is non-programmable). Nenad's solution is of course the very same, and **John Keith**'s RPL one is even shorter at essentially just one statement, **TRACE**.

Of course, the 'fun factor' in this case was to realize that the apparently very complicated task being demanded, with seemed to require inputting a whole 16-element matrix, computing its four real/complex eigenvalues, and adding them up, was solved by simply computing the *trace*, which can be done by simply adding up the elements in the main diagonal.

I specified that it should work for 4x4 matrices because the RPN stack is 4-level high and so the program would be as simple as possible, but the same approach would work for general NxN matrices, even complex-valued ones.

2 - Simpler

"Write a program to compute and output in one go all real roots of:

$$1^x + 19^x + 20^x + 51^x + 57^x + 80^x + 82^x = 2^x + 12^x + 31^x + 40^x + 69^x + 71^x + 85^x$$

accurate to at least 100 decimal places."

The 'fun' here resides in the fact that this transcendental equation happens to have *exactly 7 real roots that happen to be the natural integers from 0 to 6*, i.e.: **0, 1, 2, 3, 4, 5, 6**, so they can be computed to arbitrary precision as their 'infinite' trailing decimal places all happen to be zero.

My original solution for the HP-71B is as follows:

```
10 DEF FNF(X)=1^X+19^X+20^X+51^X+57^X+80^X+82^X-2^X-12^X-31^X-40^X-69^X-71^X-85^X
20 FOR X=-10 TO 10 @ R=FNROOT(X,X+1,FNF(FVAR)) @ IF R>=X AND R<X+1 THEN DISP R
40 NEXT X


>RUN


     0   1   2   3   4   5   6
```

which makes use of the observation that for sufficiently large negative X there can't be any more real roots because of the predominance of some terms for those arguments, and the same goes for sufficiently large positive arguments, where other terms tend to predominate. My solution above checks the reasonable range and outputs all seven roots. You can check them from the command line like this:

```
>FOR X=-1 TO 7 @ DISP X;FNF(X) @ NEXT X


  -1   .483545039294
   0   0
   1   0
   2   0
   3   0
   4   0
   5   0
   6   0
   7  -36021585500
```

Admittedly, this doesn't guarantee that there are no additional real roots apart from those seven but once computed that can be easily ascertained by looking at the graph or, ultimately, finding maxima and minima, etc, but that would defeat the 'fun' factor, which simply was the unexpected small integer roots in arithmetic progression, thus computable to 100-digit or whatever accuracy with no effort whatsoever.

What's more, the mere fact that the sums of the 0^{th}, first, second, third, fourth, fifth, sixth, and seventh powers of those two integer sets, **[1, 19, 20, 51, 57, 80, 82]** and **[2, 12, 31, 40, 69, 71, 85]**, happen to agree is in itself pretty remarkable, though far from unique: this particular equation belongs to the branch of **Multigrade Diophantine Equations** which you can get to know by visiting the link, after which I'm sure you'll agree it's a pretty enthralling topic, with many recreational aspects.

3 - Simple

"Write a program to output all machine-representable real roots of the equation:

$$(x+1)^2 - (x+2)^2 - (x+3)^2 + (x+4)^2 - (x+5)^2 + (x+6)^2 + (x+7)^2 - (x+8)^2 = 0$$

"

Here the 'fun' factor resides in the fact that despite its looks this *isn't* an equation but a very pretty *identity* in disguise, namely **0 = 0**, thus it is *independent* of the argument X and so *each and every real and complex value of X will satisfy it*.

Once this funny realization is over, however, it remains the hardly trivial task of generating and outputting *each and every machine-representable value in order*, at least in theory (running time would be utterly preposterous), which varies from model to model. My original solution for the HP-71B is the following 2-line (54-byte) program:

```
1 DESTROY ALL @ N=TRAP(UNF,2) @ N=-INF @ DISP N
2 N=NEIGHBOR(N,INF) @ DISP N @ IF N#INF THEN 2
```

```
>RUN

   -Inf
   -9.99999999999E499
   -9.99999999998E499
   -9.99999999997E499
   -9.99999999996E499
   -9.99999999995E499
      ...
   -1.00000000002E499
   -1.00000000001E499
   -1.E499
   -9.99999999999E498
   -9.99999999998E498
      ...
   -1.00000000002E-499
   -1.00000000001E-499
   -1.E-499
   -0.99999999999E-499
   -0.99999999998E-499
      ...
   -0.00000000002E-499
   -0.00000000001E-499
   -0
    0.00000000001E-499
    0.00000000002E-499
      ...
    9.99999999998E499
    9.99999999999E499
    Inf
```

which outputs in order every machine-representable value from **-Inf** to **Inf**, including such denormalized values as **-0.00000000001E-499**.

The **NEIGHBOR** function merrily provides each value in turn, and the **TRAP** setting makes sure it will return denormalized values as well, for completeness' sake. **Jean-François Garnier** also discovered the use of **NEIGHBOR** and further provided a different method suitable for conversion to most any other model. Regrettably no RPN or RPL versions were provided, not even blind conversions of J-F's routine.

The Challenge:

---

1 - Simplest

"Write a program to split the set of integers [0, 1, ..., 15] into two sets with the same number of elements such that the respective sums of the elements, the squares of the elements, and the cubes of the elements of each set are equal."

The fun factor here is that it appears tremendously difficult and/or time-consuming to find the correct split fulfilling the conditions, much more so for simple models like the HP-10C; yet given the proper approach it's actually unbelievably easy.

Egan Ford was the first to stumble upon the amazing solution, which is (again) a kind of Multigrade Diophantine Equation, as seen above, and which depends on the parity of the number of 1's in the binary representation of the set elements 0, 1, ..., 15.

Of course, the ideal HP model for "binary tasks" is the HP-16C, and here is my original *9-step* solution for it:

```
01  LBL A
02  RCL I
03   #B
04   SR
05  RCL I
06  R/S
07  DSZ
08  GTO A
09   /
```

The solution is pretty general and works for upper limits other than 15 (such as 3, 7, 31, 63, 127, etc). For **15**, it requires **DEC** mode, and **WSIZE** >= 5 in 2's-Complement mode, or **WSIZE** >= 4 in Unsigned mode (if the upper limit of the set were **31**, you would use **WSIZE** >= 6 and >= 5, respectively; similarly for larger or smaller upper limits).

To run it, simply previously store the upper limit in register **I** and execute **GSB A**: each element of the set will be displayed, in reverse numerical order, with elements belonging to the first subset being displayed *as is*, while elements belonging to the *Complementary* subset will be displayed with the **"C"** (for "Complementary") annunciator being turned on, like this:

```
15, STO I, GSB A ->  15  d  (belongs to the first set)
             R/S ->  14C d  (belongs to the Complementary set)
             R/S ->  13C d  (belongs to the Complementary set)
             R/S ->  12  d  (belongs to the first set)
                 ...
             R/S ->   2C d  (belongs to the Complementary set)
             R/S ->   1C d  (belongs to the Complementary set)
             R/S ->   0  d  (belongs to the first set)
```

the solution thus being:

```
first set         : [ 0, 3, 5, 6, 9, 10, 12, 15 ]
complementary set : [ 1, 2, 4, 7, 8, 11, 13, 14 ]
```

and the sums of their 0^{th}, first, second, and third powers are **8, 60, 620, and 7200** in both cases (the fourth powers differ, of course: 89924 and 88388, respectively).

An equivalent generalized HP-71B solution would be, for instance:

```
10 DESTROY ALL @ INPUT "K=";K @ FOR N=0 TO 2^K-1
20 S=0 @ FOR H=0 TO K-1 @ S=S+BIT(N,H) @ NEXT H
30 IF MOD(S,2) THEN DISP -N ELSE DISP N
40 NEXT N


>RUN


 K=4


    0 -1 -2  3 -4  5  6 -7 -8  9  10 -11  12 -13 -14  15
```

where positive elements belong to the first set, negative elements to the complementary set. A solution for the HP-10C can be written very easily in a few dozen steps, but I've been unable to check it on a real HP-10C or a ROM-based emulator so I won't publish it here. It isn't difficult at all to convert the above solutions, of course.

2 - Simpler

"Write a program to find two distinct non-negative sequences of 2^N elements which have the same set of pairwise sums"

The fun factor here is that the solution to this apparently totally unrelated task *is actually the same as for subchallenge 1 above !!*

Thus, the *very same routine* will produce the identical sets that, apart from having their respective sums of powers equal, as seen above, also have the additional property of resulting in the same pairwise sums.

So, all the generalized solutions above apply and if you discovered this awesome fact you would get two birds for the price of one ! :-)

My original solutions for the last three, much more substantial remaining subchallenges will follow in Part 2.

Best regards from V.

Re: My Original Solutions and Comments - Part 1 (Re: S&SMC#20)

*Message #79 Posted by **Fernando del Rey** on 21 Apr 2008, 1:48 p.m.,*
in response to message #78 by Valentin Albillo

Dear Valentín,

Just a couple of lines from someone who has dedicated some time to work on the S&SMC#20 challenges (though admittedly not enough time), without reaching anything that would be worth posting on the Forum.

I'm pretty sure there are many others anonymous readers like me, who have enjoyed the challenges and the ingenious responses from the clever people in this Forum. Even though I have not contributed back, I still fully appreciate your efforts and those of the other contributors, and wish to publicly thank you for it.

And if I may ask for a wish: why don't you make the S&SMC a once-per-year event, always on April 1st. It would make a nice tradition in this Forum.

Still, if you call it quits on #20, I must say you've done more than enough. I will look forward to your mini-challenges.

Regards, Fernando

Re: My Original Solutions and Comments - Part 1 (Re: S&SMC#20)

*Message #80 Posted by **Valentin Albillo** on 21 Apr 2008, 5:18 p.m.,*
in response to message #79 by Fernando del Rey

Hi, Fernando:

Thanks for your kind & friendly comments. Re making S&SMC's a once-per-year affair, it's actually what I've been doing for the past three years (save for the Valentine's Day Special, which was intended as training for the April's Fool one).

However, taking this last S&SMC#20 as an example, I began to research materials for it as early as January. Making a long list of suitable problems, refining it to a short list, then programming my own original solutions to the problems (including lots of test cases, optimizing, and debugging) took the remaining time till I posted the result last April 1st. In other words, a lot of free time

used for this and a lot of work. Even if once a year. And the participation, however high-level it might be, just doesn't warrant it. Thinking otherwise amounts to delusion, which I'm not prone to.

So I'll stick to mini-challenges for now. Thanks a lot for your interest and kind words, it's a pity I never got to see your solutions (or even attempts at solving my challenges). I know you are fully capable to solve them all hands down if given just a tiny amount of free time, which regrettably I also know you simply don't have.

Very Best regards from V.

Re: My Original Solutions and Comments - Part 1 (Re: S&SMC#20)

*Message #81 Posted by **Marcus von Cube, Germany** on 30 Apr 2008, 3:52 a.m.,*
in response to message #79 by Fernando del Rey

Hi Valentin,

I was hoping to find the time to tackle some of the problems you posted in S&SMC#20 but my job simply didn't leave enough time. :(

My HP-50g happily solved the prechallenge #2 and I found out about the identity in prechallenge #3. But that was it. (My knowledge about eigenvalues has faded away in the last 24 years.)

What I still don't get is the reasoning behind challenges #1 and #2. What has the number of bits to do with the sum of powers or the pairwise sums? Is this an incident you need to know about or can it be deducted in an understandable way?

Thanks again for your efforts which make me feel like an (April's) fool in most of the cases.

Marcus

Re: My Original Solutions and Comments - Part 1 (Re: S&SMC#20)

*Message #82 Posted by **Valentin Albillo** on 1 May 2008, 4:15 p.m.,*
in response to message #81 by Marcus von Cube, Germany

Hi, Marcus:

"I was hoping to find the time to tackle some of the problems you posted in S&SMC#20 but my job simply didn't leave enough time"

I fully understand you. Job is a real killer for ever having decent free time; these days I had an important deadline to meet, and all my team's been working for 30 days straight, no weekends, no rest, I went home last week at 10:30 PM each and every day, from 7:00 AM in the mornings. Were it not for the fact that I sleep so little, I'd never have time to get anything done.

"My knowledge about eigenvalues has faded away in the last 24 years"

It was a trick problem: the only required piece of knowledge was that the sum of the eigenvalues equals the sum of the main diagonal elements (i.e.: the *trace*)

"What I still don't get is the reasoning behind challenges #1 and #2. What has the number of bits to do with the sum of powers or the pairwise sums? Is this an incident you need to know about or can it be deducted in an understandable way ?"

It's also tricky. Remember this was an April's Fool challenge, which means the problems seem impossible at first sight, as if they were jokes, yet can be solved but in some weird, retorted, tricky way. You can learn all about it by checking these links:

Thue-Morse sequences (Wikipedia)

Thue-Morse sequences (Mathworld)

and simply *googling* for ***"Morse sequence"*** will report those and many extremely interesting links about it. There's also a number of books dealing with it and its awesome properties. For instance, though I only asked that Sum(x^0), Sum(x^1),Sum(x^2), and Sum(x^3) be equal for both subsets, actually the sums are also equal for *any* arbitrary 3rd-degree polynomial, i.e.:

$$\text{Sum}(2*x^3 + 0*x^2 + 0*x^1 + 8*x^0)$$

or

$$\text{Sum}(1*x^3 + 9*x^2 + 5*x^1 + 8*x^0)$$

are also equal for both subsets and matter of fact I did consider asking for some such polynomials instead of the four discrete powers from 0 to 3, to make it even more puzzling and joke-like.

"Thanks again for your efforts which make me feel like an (April's) fool in most of the cases."

That's not the idea, Marcus, but simply to bring a smile to everyone trying to solve them (the identity-based problem, for instance), and further to create a feeling of awe at the underlying wonderful mathematical facts and insights.

Thanks a lot for your interest and kind support of my efforts, and

Best regards from V.

My Original Solutions and Comments - Part 2 (Re: S&SMC#20)

*Message #83 Posted by **Valentin Albillo** on 22 Apr 2008, 9:56 a.m.,*
in response to message #78 by Valentin Albillo

Hi, all:

This is the second part of ***"S&SMC#20: My Original Solutions & Comments"*** where I'll give my original solutions (and comments) to subchallenges #3 and #4 , which are much more substantial mathematically speaking than the ones already discussed in Part 1. Let's see:

The Challenge (Continued)

---

3 - Simple

"Ms. Germain selects two integers between 1 and 100 (both excluded) and tells their sum only to Susan and their product only to Peter, both assumed to be ideally intelligent and honest students. She then asks if they can figure out what the original numbers are, and their dialogue goes as follows:

```
Peter:  I'm sorry Ms. Germain but I can't figure out the numbers right now ...
Susan:  Of course, I knew that you couldn't. Anyway, right now I can't either ...
Peter:  Thanks, Susan, because now I know them !
Susan:  You're welcome, Peter, for now I know them too !!
```

If they could, your HP can too, so write a program to find the two original numbers."

The fun factor here is that at first sight it seems that the dialogue isn't actually providing sufficient information to the participants so line 1 and 2 seem more or less a given while line 3 and 4 seem to reach a conclusion out of nowhere.

That's not so, of course, and on deeper analysis each line conveys some useful information that both participants can use to narrow the possibilities till there's but one left. Both **PeterP** and **Jean-François Garnier** provided clever analysis and careful explanations, as well as functional code, which mostly clarify the subject so I suggest you read them carefully to understand what's happening. If you still are unsure what's going on, then this excellent PDF document will make it crystal-clear, line for line, and as rigorously as it needs to be. You may also enjoy having a look at all these interesting variations, many of which are as challenging as the one discussed here, if not more.

Writing a program to solve this problem depends on the amount of hard thinking you do before. On one extreme, you do everything on your own, then your program simply prints out the answer. On the other extreme, you do next to no thinking and use a brute-force approach to test all possible cases. The correct approach is an intermediate one but weighted to favor the program doing much more work than you, as that's the idea behind writing code in the first place. My original solution following this criterium is this small HP-71B program:

```
 10 DESTROY ALL @ OPTION BASE 1 @ DIM G$[195]
 20 M=0 @ FOR S=5 TO 53 STEP 2 @ FOR X=2 TO S-2 @ IF FNF(X*(S-X)) THEN 40
 30 NEXT X @ M=M+1 @ G$=G$&CHR$(S)
 40 NEXT S @ FOR N=1 TO M @ S=NUM(G$[N]) @ P=0
 50 FOR X=2 TO S-X @ Q=FNR(X*(S-X)) @ P=P+Q @ IF Q THEN T=X
 60 NEXT X @ IF P#2 THEN 80 ELSE P=T*(S-T) @ A=(S-SQR(S*S-4*P))/2 @ B=S-A
 70 DISP "Sum=";S;", Prod=";P;"=> X=";A;", Y=";B @ END
 80 NEXT N @ END
 90 DEF FNF(X) @ R=0 @ FOR I=2 TO MIN(X DIV 2,99) @ IF MOD(X,I) THEN 110
100 IF R=2 THEN FNF=0 @ END ELSE R=R+1
110 NEXT I @ FNF=1
120 DEF FNR(X) @ K=0 @ FOR U=2 TO MIN(X DIV 2,99) @ IF MOD(X,U) THEN 140
130 IF NOT FNF(X) THEN K=K+(POS(G$,CHR$(U+X DIV U))#0)
140 NEXT U @ FNR=K=2


>RUN


   Sum= 17 , Prod= 52 => X= 4 , Y= 13  ( 0.21 seconds under Emu71, some 50 seconds on a physical HP-71B)
```

which makes use of my previous deduction that the sum must be odd and less than 54, as confirmed by the PDF document. If I hadn't reached that conclusion, line 20 would have been instead:

```
20 M=0 @ FOR S=4 TO 198 @ FOR X=2 TO S-2 @ IF FNF(X*(S-X)) THEN 40
```

where only the outer **FOR** changes to allow for the sum being anything between **4** (=2+2) and **198** (=99+99), both included. This produces the exact same solution, but takes 12 times longer. In both cases, note the use of the string function **POS** to quickly detect whether a specific numeric value (here converted to a character) appears in an array (actually stored in string form): this is faster than using a loop to look for it.

4 - Less Simple

"Write a program to, given a set of numbers, find a nonzero constant such that if you multiply each element by it, the set is now entirely composed of perfect powers."

Both **Steve Perkins** and **PeterP** did very well with this problem, with Steve providing an early but simple solution which indeed technically solved the problem but resulted in highly non-optimized solutions, many orders of magnitude larger than necessary. His second attempt greatly improved on the first one, but still fell short of what was needed, which required a different approach. Nevertheless, his program is very small and his approach very clever so he certainly qualifies as a successful solver.

PeterP, on the other hand, did try for the more sophisticated approach, which involves obtaining the prime factorization of each number in the set, then solving a series of linear congruences to determine the set of exponents for each prime factor which will form part of the multiplicative constant. Upon multiplication, the exponents add up to the corresponding ones on each factor to conform perfect powers.

This is also the approach I originally took, with some variations, and this 23-line (849 byte) program is my original solution for the HP-71B:

```
 1 DESTROY ALL @ OPTION BASE 1 @ INPUT "#=";N @ Z=4*N
 2 DIM A(N),F(N,Z),E(N,Z),P(Z),H(Z),U(Z),V(Z),W(Z)
 3 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71
 4 READ H @ MAT INPUT A @ FOR I=1 TO N @ K=A(I) @ D=2 @ J=0
 5 IF K=1 THEN 8 ELSE IF MOD(K,D) THEN D=3 ELSE GOSUB 22 @ K=K DIV D @ GOTO 5
 6 FOR D=D TO SQR(K) STEP 2 @ IF NOT MOD(K,D) THEN GOSUB 22 @ K=K DIV D @ GOTO 6
 7 NEXT D @ D=K @ GOSUB 22
 8 NEXT I @ K=1 @ FOR I=1 TO N @ FOR J=1 TO Z @ M=F(I,J) @ IF NOT M THEN 12
 9 FOR L=1 TO K-1 @ IF P(L)=M THEN 11
10 NEXT L @ P(K)=F(I,J) @ K=K+1
11 NEXT J
12 NEXT I @ K=K-1 @ DIM X(N,K) @ FOR I=1 TO N @ FOR J=1 TO Z @ IF NOT F(I,J) THEN 15
13 FOR L=1 TO K @ IF F(I,J)=P(L) THEN X(I,L)=E(I,J) @ GOTO 15
14 NEXT L
15 NEXT J @ NEXT I @ M=1 @ FOR I=1 TO N @ M=M*H(I) @ NEXT I
16 FOR I=1 TO N @ U(I)=M/H(I) @ NEXT I @ FOR I=1 TO N @ V(I)=MOD(U(I),H(I)) @ NEXT I
17 FOR I=1 TO N @ IF V(I)#1 THEN V(I)=(H(I)+1)/V(I)
18 NEXT I @ FOR I=1 TO K @ FOR J=1 TO N @ W(I)=W(I)-V(J)*U(J)*X(J,I) @ NEXT J
19 W(I)=MOD(W(I),M) @ NEXT I
20 DIM S$[80] @ S$="" @ FOR I=1 TO K @ S$=S$&STR$(P(I))&"^"&STR$(W(I))&"*"
21 NEXT I @ S$[LEN(S$)]="" @ DISP S$;" =";VAL(S$) @ END
22 IF NOT J THEN 23 ELSE IF F(I,J)=D THEN E(I,J)=E(I,J)+1 @ RETURN
23 J=J+1 @ F(I,J)=D @ E(I,J)=1 @ RETURN
```

which *doesn't* make any use of the **JPC ROM** as PeterP's extensively does (it wasn't allowed by the rules but this being the last S&SMC I'll make an exception for PeterP's sake).

Upon running, it first asks for the number of elements in the set, then for the elements themselves, and outputs an exact representation of the multiplicative constant, plus its numeric value if within range (else *Overflow*). For instance:

```
>RUN


 # = 3
 A(1)?  2,3,4


   2^3*3^20 = 27894275208
```

and we can check that

$$2 * 27894275208 = 55788550416 = 236196^2$$
$$3 * 27894275208 = 83682825624 = 4374^3$$
$$4 * 27894275208 = 111577100832 = 162^5$$

thus

$$[\ 2,\ 3,\ 4\ ] * 27894275208 = [\ 236196^2,\ 4374^3,\ 162^5\ ]$$

so it does pass muster. This is a table of the respective solutions achieved with my program (some 0^{th} powers have been supressed for clarity), and PeterP's (Steve Perkins' are *many* orders of magnitude greater in most cases):

```
     Set                 My original solution                 PeterP's
-------------------------------------------------------------------------------------------
 [ 41, 67 ]              41^3 * 67^2         = 309386369               same
 [ 1776, 2008 ]          3^3 * 37^3 * 251^2  = 86162120631             same
 [ 2007, 2008 ]          223^3 * 251^2       = 698653810567            same
 [ 1958, 2008, 50 ]      2^9 * 11^15 * 89^15 * 251^20 * 5^18      2^17 * 11^20 * 89^20 * 251^24
                                             = 1.399..E108          = 3.35..E122
 [ 2, 3, 4]              2^3*3^20            = 27894275208             same


 [ 3, 4, 5 ]             3^15 * 2^10 * 5^24     = 8.75..E26            same
 [ 2, 3, 4, 5 ]          2^63 * 3^140 * 5^^90   = 4.66..E148           same
 [ 2, 3, 4, 5, 6]        2^483 * 3^560 * 5^1980 = Overflow             same


 [ 10, 11, 12, 15, 16 ]  2^2163 * 5^825 * 11^770 * 3^594   = Overflow  same
 [ 71, 42, 15 ]          71^15 * 2^20 * 3^14 * 7^20 * 5^24 = 1.4..E74  same
-------------------------------------------------------------------------------------------
```

As you can see, PeterP's solutions coincide with mine except for the case of the **Halphacentaury set** (named not after *"alpha centauri"* but after *"half a century"* :-) **[1958, 2008, 50]**, and *both are correct*, only mine is *14 orders of magnitude smaller* which may seem a lot but when exponents over 100 are involved it isn't that much really.

By the way, my program (and obviously PeterP's as well) does produce a fairly optimized solution, in the sense that the output constant is reasonably small but it isn't guaranteed to be the *smallest* possible. Matter of fact, simply inputting the set elements in a *different order* usually produces different results, which can be smaller than the one you get with the original order. Just for instance, using my program we get:

```
[ 2, 3, 4]  ->   2^3 * 3^20  = 27894275208
```

but

```
[ 3, 4, 2]  ->   3^15 * 2^4  =   229582512
```

which is two orders of magnitude (i.e., around 100) times smaller. Thus, there's room for further optimization and a very simple-minded approach would be to just find constants for all permutations of the set elements and pick the smallest one. This is horribly inefficient and prohibitive for large sets but for sets of 2,3,4,5 (N) elements you'd need to choose among 2,6,24,120 (N!) values for the minimum constant, which is just about doable in a pinch.

Further improvements of my program could include:

- general code optimization; the code as posted is about the first working version I produced, and would probably benefit from the usual optimization techniques which I simply couldn't try for lack of free time.
- The output routine is quite simple and does not screen for 0^{th} powers (i.e.: 2^0, say) which do no harm but are unsightly, so it would be proper to include code to ignore them.
- some additional code to ensure the smallest possible constant would be nice.
- as written, it will work with sets up to 5 elements; to be able to handle bigger sets additional primes should be added to the ones included in the **DATA** statement at line 3.

My original solution for the last, most difficult remaining subchallenge (#5) will follow in Part 3 within 24 hours, plus I'll add further comments and final notes for all of them, in order to gracefully conclude the series.

Best regards from V.

Edited to correct a typo

Edited: 22 Apr 2008, 10:12 a.m.

My Original Solutions and Comments - Part 3 (Re: S&SMC#20) [LONG and I mean it]
*Message #84 Posted by **Valentin Albillo** on 26 Apr 2008, 10:47 p.m.,*
in response to message #78 by Valentin Albillo

Hi, all:

This is **Part 3** of ***"S&SMC#20: My Original Solutions & Comments"*** which at long last concludes with my solution for the very last (and most difficult) subchallenge **#5**. Several of you bravely attempted to conquer it with various degrees of success: both **Eamonn** and **Egan Ford** posted actual results and finally Egan posted **HP-71B** code. However, their results do *not* fully agree with mine in all cases for whatever reasons. That said, let's proceed:

The Challenge (concluded)

---

5 - Least Simple

"The sequence defined by

$$(n + 1) * x_{n+1} = n*x_n^k + x_n$$

with $\mathbf{k = 2}$ *and* $\mathbf{x_1 = 3}$*, continues with* $x_2 = \mathbf{6}$, $x_3 = \mathbf{16}$, $x_4 = \mathbf{76}$, $x_5 = \mathbf{1216}$*, and* $x_6 = \mathbf{247456}$*, all of which are nicely integer. Regrettably,* $x_7 = \mathbf{8747993810.2857}$*..., which is not. Write a program which, given* $\mathbf{k}$ *and* $\mathbf{x_1}$*, both integer and* $\mathbf{> 1}$ *, finds and outputs both the index of the first element of the sequence which fails to be an integer, if any, as well as its nonzero fractional part."*

The fun factor here is that the task would be absolutely trivial were it not for the fact that the terms of the sequence grow *unbelievably enormous extremely quickly* for most values of **k** and $\mathbf{x_1}$. Just for instance, for the "simplest" case **k=2** and $\mathbf{x_1}$**=2**, the sequence merrily goes:

$x_2 = 3$, $x_3 = 5$, $x_4 = 10$, $x_5 = 28$, $x_6 = 154$, $x_7 = 3520$, $x_8 = 1551880$,
$x_9 = 267593772160$, $x_{10} = 7160642690122633501504$, ...

and so on till the last integer element at index **42**, which has about *89,288,343,500 digits* !!. The next element, x_{43}, is *not* integer and, using the asymptotic formula:

$$x_n = (n + 2 - 1/n + 4/n^2 - 21/n^3 + 138/n^4 - 1091/n^5 + ...) * C^{2^n}$$

where **C = 1.04783144758**..., approximately evaluates to:

$$x_{43} = \mathbf{5.41 * 10^{178,485,291,567}}$$

but a number more than ***178 billion digits long*** isn't something that any current computer can handle, let alone handheld programmable calculators, not even HP's. Now, if this is the case for the quite modest **k=2**, $\mathbf{x_1=2}$, and $\mathbf{x_{43}}$, just try to imagine the humongous *size* of $\mathbf{x_{601}}$ (!!), the first non-integer element for **k=3**, $\mathbf{x_1=11}$. Even merely computing its exact *size* is far from trivial, let alone the *element* itself !

Consider also the extremely *disconcerting* mathematical nature of what's happening here: a simple recurrence which unfailingly produces a sequence of integer elements for the first 600 cases suddenly and most unexpectedly results in a non-integer one for case 601, and the same goes for other values of the parameters.

What to do ? Well, I see three possible approaches. On one extreme you could try using the *full multiprecision route* and proceed to compute each element till the first non-integer one is found. This may succeed for a few values of **k** and $\mathbf{x_1}$ (**k=2**, $\mathbf{x_1=3}$ or **k=8**, $\mathbf{x_1=10}$, for instance) but for the most part, as seen in the case of **k=2**, $\mathbf{x_1=2}$, it's a *totally hopeless* dead end, now and in the foreseeable technological future.

The other extreme would be using *as little multiprecision as possible or even none at all* and try to cope using *modular arithmetic* and solving *congruences* to go forward seeking for the elusive non-integer element. This is mostly the route Eamonn and Egan followed, but though they succeeded in the majority of cases at minimal computing cost, they found some recalcitrant ones which, at the time of writing this, they haven't completely succeeded in ironing out.

My original solution is an intermediate one. I don't use the full multiprecision approach, as I quickly saw the task was impossible, but I'm using *modular arithmetic to a sufficiently high precision* (up to 500 digits for the version I'm posting here) that the algorithm is *straightforward* and *no ambiguities* arise at any point, just a very simple loop till either the first non-integer element is found, or the maximum precision is reached, or time becomes excessive, whichever comes first.

The only problem for me is that I wanted to implement my solution for the **HP-71B** which, regrettably, *lacks* the multiprecision integer features of advanced RPL models. However, this only means that besides the simple code to implement the main algorithm *(6 lines)*, I needed to implement a multiprecision *"library"* which would allow calls from the main program to perform the requested multiprecision operations on demand.

This isn't particularly easy, most specially as it must both run *fast* and be as small and compact as possible since this is a simple ***S&S Math Challenge***, not a full-fledged article on how to implement multiprecision libraries in the **HP-71B**. With that in mind, I created the following original solution for the **HP-71B**:

Main program (just *6 lines* of **HP-71B** BASIC code implementing the base algorithm):

```
10 DESTROY ALL @ OPTION BASE 1 @ D=200 DIV 6+1 @ DIM F(D),X(D),Y(D),Z(D)
20 INPUT "K,X1=";K,X1 @ SETTIME 0 @ FOR N=10 TO 2000 STEP 10 @ CALL MFAC(N,F)
30 DISP N; @ CALL N2M(X1,Y) @ FOR I=2 TO N @ MAT Z=Y @ CALL MDIVK(Z,I-1)
40 CALL MPOW(Z,K,X) @ CALL MSUM(Y,X,Z) @ CALL MMOD(Z,F,Y) @ CALL MMODK(Y,I,M)
50 IF M THEN DISP @ DISP I;M/I;TIME @ END ALL
60 NEXT I @ NEXT N
```

As **HP-71B**'s BASIC lacks any multiprecision operations, integer or otherwise, the main program simply calls the relevant library functions (you'll find a reasonably detailed description of the library in **Apendix A** below). The main program is executing the equivalent of the following *pseudo-code*:

```
dimension multiprecision variables F, X, Y, and Z to hold a certain number of digits (200 above)
ask the user for K and X1
for values of N from 10 to 2000 (both included) and counting in steps of 10, do:
    assign the factorial of N to F
    assign the value of X1 to Y
    for values of I from 2 to N, both included, do
        assign the value of Y to Z
        integer-divide Z by I-1
        assign the value of Z raised to the kth power to X
        assign the value of X + Y to Z
        assign the value of Z Modulus F to Y
        assign the value of Y Modulus I to M
        If M is not zero, then we've found the first non-integer element, so
           output index and fractional part and terminate
    loop to process the next value of I
 loop to process the next value of N
```

My original program is an implementation extensively relying in full multiprecision modular arithmetic, and taking into account the ideas discussed in the book ***"Mainly Natural Numbers"*** which you can freely download in PDF format from here. This particular problem is featured in **Chapter IV:** ***"Some Sequences of Large Integers"***, *pp. 32-37.*, though the notation there differs slightly from mine; for instance my case **k=2**, $\mathbf{x_1=2}$, appears there as **m=1**, $\mathbf{x_1=2}$, etc. You're advised to read it for the rigorous analysis and relevant details, as space (and notational difficulty inherent to text-based HTML) does not permit to discuss the subject at lenght here.

Now this is a sample run for the case **k=2**, $\mathbf{x_1=41}$:

```
>RUN

  k,x1=2,41

    10  20 ... ->   17  .235294117647  1.53
```

where the *running index* is shown at steps of 10 while the computation is in progress, then the *index of the first non-integer element* is output (**17**), as well as its *fractional part* (**0.235294117647**) and the *time* it took to find it (**1.53** seconds under Emu71, a little over 6 min. in a physical HP-71B).

Note:

The default precision listed above in line 10 is 200 digits, which is more than enough for all the cases I asked except the two final, extra ones which require about 360 and 415 digits, respectively. If you get an error (normally "Subscript") while checking some case, just increase the precision from 200 to some suitable higher value less than 500 in line 10 above.

Running my program for the asked values, one gets the following results and times (for Emu71 @ 2.4 Ghz, single Pentium 4 CPU with 512 Mb RAM under Windows XP Home Edition 2002; for a physical HP-71B, multiply the given times by roughly 250):

k	x_1	index	#digits	fractional part	time
2	2	43	200	0.558139534884	19.10
2	3	7	200	0.285714285714	0.26

```
 2    15    67     200     0.268656716418     77.60
 2    41    17     200     0.235294117647      1.53
 2    42    59     200     0.559322033898     44.40
 2    50    34     200     0.117647058824     10.33
 2    67    51     200     0.411764705882     41.56
 2    71    17     200     0.235294117647      1.58


 2  1958    17     200     0.235294117647      1.64
 2  1992    17             0.235294117647      1.64
 2  2008    61     200     0.983606557377     77.10


 2    99   103     360     0.650485436893    593.41
 3     2    89     415     0.460674157303    686.60
```

and further this is a short table of extra results for comparison purposes with the book and/or your own programs (140 digits are more than enough so edit line 10 accordingly):

```
  k    x1  index  #digits   fractional part    time
-----------------------------------------------------


  3     6    31     140     0.645161290323     16.40
  4     7    13     140     0.692307692308      2.02
  6    11    19     140     0.263157894737      5.19
  8     3     7     140     0.285714285714      0.60
  8     8    17     140     0.588235294118      6.01
```

Finally, this is an extensive table of results obtained with a different version of my *ad-hoc* library, which allows for much greater number of digits but takes at least one order of magnitude longer (the posted version can only handle slightly less than 500 digits, being limited by the exponent range of the built-in **REAL** precision, but it's smaller and much faster, and thus better suited for the cases asked in this subchallenge).

For each assorted $\mathbf{x_1}$ and **k**, two numbers are given: the ***index*** of the first non-integer element, and the *approximate number of digits* necessary to carry out the computations, which will appear as **"?"** if the program could not reach far enough before I gave up due to timing considerations. Some of these results appear in the book I mentioned (and fully coincide with them), but the rest I've never seen published before so this can be considered *original research publicly appearing for the first time in the* ***World Wide Web*** and so independent corroboration would be most welcome.

```
  x1     K=2         K=3         K=4         K=5         K=6         K=7         K=8         K=9         K=10
-------------------------------------------------------------------------------------------------------------
   2  43  103    89   395    97   598   214  2024   19   95   239     ?   37   331    79  1036   83  1222
   3   7    6    89   403    17    54    43   255   83  733   191  2464    7    20   127  1901   31   323
   4  17   27    89   403    23    84   139  1182   13   52   359     ?   23   167   158  2502   41   469
   5  34   74    89   402    97   599   107   845   19   95   419     ?   37   331    79  1030   83  1224
   6  17   27    31    97   149  1032   269     ?   13   47   127  1477   23   169   103  1449   71  1000
   7  17   26   151   788    13    35   107   849   37  250   127  1475   37   332   103  1457   83  1226
   8  51  128    79   343    13    35   214  2025   13   50   239     ?   17   107   163     ?   71  1000
   9  17   25    89   402    83   491   139  1180   37  247   191  2464   23   167   103  1453   23   211
  10   7    6    79   343    23    84   251  2460  347    ?   239     ?    7    19   163     ?   41   476
  11  34   73   601     ?    13    34   107   848   19   88   461     ?   37   329    79  1035   31   323
  12  17   27   197  1095    97   600     ?     ?   37  248   191  2436   17   102    79  1034   41   478
  13  17   26   151   788    23    85     ?     ?   37  248   127  1480   37   327   158  2495   31   323
```

```
  14  43  103   158   834    67   369   139  1184   37  245   191  2462   23   161   158  2503   41   473
  15  67  186   197  1097   173  1244   139  1182   37  249     ?     ?   37   332   127  1902   31   322


  41  17   25    79   346    17    52   107   851   83  736   127  1479   23   168    79  1026   23   209
  42  59  157   151   789    23    79   251  2462   37  244   191  2463   17   105     ?     ?   31   324
  50  34   74   193  1070    13    35   107   848   37  248   191  2463   23   169    79  1028   41   478
  67  51  129    89   401    97   599   251  2463   37  250   191  2462   23   164    79  1031   83  1227
  71  17   26    79   344    17    53   107   850   13   52   127  1477   46   446     ?     ?   89  1342
  99 103  324    31    97    13    34   139  1182   13   44     ?     ?   37   332   151  2355   41   465


1958  17   27   158   834    13    34   251  2461   13   47     ?     ?  149  2066     ?     ?   41   474
1992  17   27   197  1098    17    53   107   850  109 1040     ?     ?   23   168   103  1456   89  1342
2008  61  164   229  1321   167  1191     ?     ?   13   52   191  2462   37   327     ?     ?   31   322
```

That's all. This ***"Short & Sweet Math Challenges"*** series exclusive to the MoHP ends its run now. The very first **S&SMC#1** was posted in May, 27th 2002 at 9:59 a.m., so it's been nearly *six years* sharing what I hope has been interesting and rarely-seen math ideas and highly creative programming techniques for an incredible variety of HP (and non-HP) calc (and non-calc) models.

Thanks to all of you for your unfailing interest and for your extremely worthy contributions, it's been a pleasure and a continued source of learning and awe to me.

Very Best Regards From V.

Appendix A: The multiprecision library

I'll briefly discuss here the small *(just 32 lines of code)* **"*Ad-hoc* multiprecision library"** I've implemented specifically for this particular subchallenge (the following listing is numbered as if to be included in the same file as the main program without line numbers conflicting, but could reside in a separate program file in RAM as well, if intended to be used by other programs).

***Ad-hoc* multiprecision library**:

```
 70 !
 80 SUB MMUL(A(),B(),C()) @ MAT C=ZER @ CALL MTOP(A,U) @ CALL MTOP(B,V)
 90 K=1000000 @ FOR I=1 TO V @ M=B(I) @ IF NOT M THEN 120 ELSE L=I
100 FOR J=1 TO U @ N=A(J) @ IF N THEN P=C(L)+M*N @ C(L)=RMD(P,K) @ C(L+1)=C(L+1)+P DIV K
110 L=L+1 @ NEXT J
120 NEXT I
130 SUB MPOW(A(),N,C()) @ IF NOT N THEN MAT C=ZER @ C(1)=1 @ END ELSE MAT C=A
140 IF N=1 THEN END ELSE DIM B(UBND(C,1)) @ U=LOG2(N) @ IF FP(U) THEN 160
150 FOR I=1 TO U @ MAT B=C @ CALL MMUL(B,B,C) @ NEXT I @ END
160 FOR I=2 TO N @ MAT B=C @ CALL MMUL(B,A,C) @ NEXT I
170 SUB MSUM(A(),B(),C()) @ MAT C=A+B @ CALL MNOR(C)
180 SUB MSBI(A(),B()) @ MAT A=A-B @ CALL MNOR(A)
190 SUB MMODK(A(),K,M) @ DIM B(UBND(A,1)) @ MAT B=A @ M=K @ CALL MDIVK(B,M)
200 SUB MDIVK(A(),K) @ FOR I=UBND(A,1) TO 2 STEP -1 @ P=A(I) @ IF P=0 THEN 220
210 A(I)=P DIV K @ A(I-1)=A(I-1)+MOD(P,K)*1000000
220 NEXT I @ P=A(1) @ A(1)=P DIV K @ K=MOD(P,K)
230 SUB MNOR(A()) @ FOR I=1 TO UBND(A,1) @ P=A(I)
240 IF P>999999 THEN A(I)=RMD(P,1000000) @ A(I+1)=A(I+1)+P DIV 1000000
250 IF P<0 THEN A(I)=P+1000000 @ A(I+1)=A(I+1)-1
260 NEXT I
270 SUB MTOP(A(),U) @ FOR U=UBND(A,1) TO 2 STEP -1 @ IF A(U) THEN END
```

```
280 NEXT U
290 SUB M2N(A(),N) @ N=0 @ FOR I=UBND(A,1) TO 1 STEP -1 @ N=N*1000000+A(I)
300 IF N>1E12 THEN N=SCALE10(N,6*I-6) @ END
310 NEXT I
320 SUB N2M(N,A()) @ MAT A=ZER @ I=0 @ M=N
330 IF M THEN I=I+1 @ A(I)=MOD(M,1000000) @ M=M DIV 1000000 @ GOTO 330
340 SUB MMOD(A(),B(),C()) @ U=UBND(A,1) @ DIM X(U),Z(U) @ CALL M2N(B,V) @ MAT C=A
350 CALL M2N(C,U) @ IF U<V THEN END
360 CALL N2M(MAX(INT(NEIGHBOR(NEIGHBOR(NEIGHBOR(NEIGHBOR(NEIGHBOR(U,0),0),0),0),0)/V),1),X)
370 CALL MMUL(X,B,Z) @ CALL MSBI(C,Z) @ GOTO 350
380 SUB MFAC(N,A()) @ CALL N2M(FACT(MIN(N,17)),A) @ IF N<18 THEN END
390 FOR I=18 TO N @ MAT A=(I)*A @ CALL MNOR(A) @ NEXT I
```

"Ad-hoc library" in this context means that it's tailored to do what the main program needs to do and absolutely nothing more, so it's mainly useful for the present purpose and not general-purpose by any stretch of the imagination. Thus, it has the following *disclaimer* attached:

- *the function set it's not complete or symmetric*: only the indispensable functionality for the task at hand is included so many unneeded operations aren't implemented at all such as integer division of two multiprecision arguments, for instance, while others are implemented only for a multiprecision argument and a single precision one. Also, some of the operations return the result in a variable different from the arguments while others are done in place. Note in particular that no routines to *input* or *output* the multiprecision values are provided.
- *it's not optimized*: all arguments are operated upon at maximum precision (say 200-digit precision), even if the actual value stored in the multiprecision variable is only 10 digits long.
- *it has no error checking at all*: most errors will result in a hard, *untrapped* error which will *stop* program execution. Also, debugging and checking hasn't been exhaustive so errors may have crept in.
- *only positive integers are supported*: any negative intermediate or final results arising will result in a hard (untrapped) error.

Most of these shortcomings are the result of designing it to run fast, be small in size, and most importantly, take as little of my scarce free time as possible to develop and debug. On the positive side, you'll find that:

- most operations are *optimized for speed* by using assembly-language *matrix operations*, as well as one or two neat tricks.
- as many as *a full dozen routines* are provided, all of them callable from your own programs (use them at your own risk).

Available calls reference:

Note: In what follows,

- *Multiprecision variables* can hold multiprecision positive integer values up to slightly less than 500 digits long (approximately), depending on the **DIM**ension of the underlying arrays. Each of them should be **DIM**ensioned (with **OPTION BASE 1** in effect) as a **REAL** precision array with **(N DIV 6)+1** elements where **N** is the *maximum number of digits*, as each array element will hold up to **6 digits**.
- *Single-precision variables* can hold positive integers up to 6 digits long.
- **REAL** variables can hold any positive integer value within the range of built-in **REAL** precision.

```
SUB MMUL(A(),B(),C())  { Multiprecision MULtiplication }
```

Takes two multiprecision variables **A** and **B** and returns their *product* in the multiprecision variable **C**. **A** and **B** are unaffected.

SUB MPOW(A(),N,C()) *{ Multiprecision **POW**er }*

Takes a multiprecision variable **A** and a single-precision value **N** and returns the value of the *power* $\mathbf{A}^{\mathbf{N}}$ in the multiprecision variable **C**. **A** and **N** are unaffected.

SUB MSUM(A(),B(),C()) *{ Multiprecision **SUM** }*

Takes two multiprecision variables **A** and **B** and returns their *sum* in the multiprecision variable **C**. **A** and **B** are unaffected.

SUB MSBI(A(),B()) *{ Multiprecision **SuB**traction **I**n-place }*

Takes two multiprecision variables **A** and **B** and returns their *difference* in the multiprecision variable **A** (subtraction in place). **B** is unaffected.

SUB MMODK(A(),K,M) *{ Multiprecision **MOD**ulus single-precision }*

Takes a multiprecision variable **A** and a single-precision value **K** and returns the single-precision value of **A** *modulus* **K** in the single-precision variable **M**, which must be passed by reference (or else won't return a thing in it). **A** and **K** are unaffected.

SUB MDIVK(A(),K) *{ Multiprecision **DIV** (and modulus) single-precision }*

Takes a multiprecision variable **A** and a single-precision variable **K** and returns the multiprecision value of the *integer division operation* **A DIV K** in the multiprecision variable **A** (i.e., in place) as well as the single-precision value of **A** modulus **K** in the single-precision variable **K** itself, which must be passed by reference. If you *don't* want or need the modulus and want **K** to remain unaffected you can simply pass **K** *by value* instead.

SUB MNOR(A()) *{ Multiprecision **NOR**malization }*

Internal use only. *Normalizes* a multiprecision variable **A** in place

SUB MTOP(A(),U) *{ Multiprecision **TOP** }*

Internal use only, though it can be useful for printing multiprecision variables, etc. Returns in the single-precision variable **U** the index of the *topmost non-zero element* of the multiprecision variable **A**. Variable **U** must be passed by reference. Variable **A** is unaffected.

SUB M2N(A(),N) *{ Multiprecision **to** real-precision **N**umeric variable }*

Returns in the **REAL** precision variable **N** the approximate value of the multiprecision variable **A** (up to 12-digit mantissa and exponent up to 499). Variable **N** must be passed by reference. Variable **A** is unaffected.

SUB N2M(N,A()) *{ **N**umeric Real-precision variable **to** Multiprecision }*

Assigns the value of the **REAL** precision variable **N** to the multiprecision variable **A**. Variable **N** is unaffected.

SUB MMOD(A(),B(),C()) *{ Multiprecision **MOD**ulus }*

Takes two multiprecision variables **A** and **B** and returns the multiprecision value of **A** *modulus* **B** in the multiprecision variable **C**. **A** and **B** are unaffected.

SUB MFAC(N,A()) *{ Multiprecision **FAC**torial }*

Takes the single-precision value **N** and returns its *factorial* in the multiprecision variable **A**. **N** is unaffected.

Sample use:

Let's say we need to compute the following:

$$69!\ \mathrm{MOD}\ 13^{68} = 161707119156747337147933149666645422128978599226831537632782504385470771962$$

This is how we would proceed (from the keyboard in this case):

1) **DIM**ension several multiprecision variables capable of holding up to **100** digits (actually 102):

```
> DESTROY ALL @ OPTION BASE 1 @ P=100 DIV 6+1 @ DIM F(P),A(P),B(P)
```

2) Make the necessary calls to the multiprecision library in order to perform the computation:

```
> CALL MFAC(69,F)          { compute the factorial of 69 }
    @ CALL N2M(13,A)       { assign the value 13 to multiprecision variable A }
    @ CALL MPOW(A,68,B)    { raise A (13) to the 68th power and assign the result to B }
    @ CALL MMOD(F,B,A)     { compute the modulus operation and assign the result to A }
```

3) Output the multiprecision result (watch your **WIDTH** setting):

```
> FOR I=P TO 1 STEP -1 @ DISP A(I); @ NEXT I @ DISP

      0 0 0 0 161 707119 156747 337147 933149 666645 422128 978599 226831 537632 782504 385470 771962
```

which is the correct result as shown above, disregarding the non-significant 0's at the beginning.

Re: My Original Solutions and Comments - Part 3 (Re: S&SMC#20) [LONG and I mean it]

*Message #85 Posted by **Egan Ford** on 29 Apr 2008, 3:10 p.m.,*
in response to message #84 by Valentin Albillo

Valentin, et al,

I have not given up on finding a solution without the use of multiprecision libraries or methods. Hopefully in time I will find one, but this thread will be long archived, so I'll post my summary now.

I've written about 20 different programs in 4 different languages with various levels of success, but no single program finds all cases.

To show that a calculator can solve even the most difficult case presented (k=3, x_1=11) I created an HPGCC2 version of Valentin's method for the 50g with the following results:

```
running...10 20 30 40 50 60 70 80
 90 100 110

k:2  x1:99  index: 103
remainder: 0.650485436893
f: 178
x: 354
y: 178
z: 354
m: 2
time(s): 6
```

```
running...10 20 30 40 50 60 70 80
 90

k:3  x1:2  index: 89
remainder: 0.460674157303
f: 139
x: 409
y: 137
z: 409
m: 2
time(s): 5
```

k=2,x_1=99 and k=3,x_1=2 on my 50g outperform EMU71+PC by a wide margin. :-)

Animation for the k=3,x_1=11 case (replays every 10 seconds):

```
running...10
```

```
 340 350 360 370 380 390 400 410
420 430 440 450 460 470 480 490 5
00 510 520 530 540 550 560 570 58
0 590 600 610

k:3  x1:11  index: 601
remainder: 0.672212978369
f: 1437
x: 4298
y: 1437
z: 4298
m: 4
time(s): 1791
```

The f,x,y,z,m labels display the length in digits.

The HPGCC2 source based on Valentin's algorithm is located at the end of this post. If any want the binary please email me. If any want to compile this or any of my C code for themselves, then read my HOWTO at: http://sense.net/~egan/hpgcc/. Two multiprecision libraries are included with this HOWTO.

My best 71B BASIC program is only capable of finding prime indexes, but it is significantly faster than using multiprecision.

EMU71 results:

```
>RUN
k=3
x1=2
k= 3  x1=   2 INDEX= 89 FRAC= .460674157303
TIME(S): 1.75

>RUN
k=2
x1=99
k= 2  x1=  99 INDEX=103 FRAC= .650485436893
TIME(S): 1.47

>RUN
k=3
x1=11
k= 3  x1=  11 INDEX=601 FRAC= .672212978369
TIME(S): 451.12
```

My BASIC program compiled with FreeBASIC can find k=25,x_1=9 (index=1451, special case (prime)) in seconds, but on the same PC with compiled C using Valentin's multiprecision method it takes about 1/2 a day. I must find a non-multiprecision method that works for all cases if one exists. :-)

Valentin,

Thank you for all your time and effort taken to create your many challenges. They do challenge me and I am sorry to see them go. I can only hope that the interest in your mini-challenges will encourage you to grant this community another SSMC next April Fools Day.

Code:

71B BASIC version (fails with composite indexes):

```
10 DESTROY ALL @ OPTION BASE 0
20 INPUT "k=";K @ INPUT "x1=";X1 @ T=TIME @ CALL DOIT(K,X1,I,R)
30 DISP "k=";K;" x1=";
40 DISP USING "4d,7a,3d,6a,d.12d";X1," INDEX=",I," FRAC=",R/I
50 DISP "TIME(S):";TIME-T
60 SUB DOIT(K,X1,I,R)
70 DIM P(1000) @ P(0)=2 @ L=1 @ Q=2
80 'START': X=X1 @ CALL NPRIME(Q,P(),L)
90 FOR N=1 TO Q
100 X=MOD(X,Q) @ T=X
110 FOR J=1 TO K-2 @ T=T*X @ T=MOD(T,Q) @ NEXT J
120 Y=MOD(X*(T+N),Q)
130 IF Y=0 THEN GOTO 'START'
140 FOR Z=1 TO Q
150 IF MOD(Z*(N+1),Q)=Y THEN X=Z @ GOTO 180
160 NEXT Z
170 I=N+1 @ R=Y @ GOTO 190
180 NEXT N
190 END SUB
200 SUB NPRIME(Q,P(),L)
210 J=P(L-1)
220 J=J+1
230 FOR I=0 TO L-1
240 IF MOD(J,P(I))=0 THEN 220
250 NEXT I
260 Q=J @ L=L+1 @ P(L-1)=Q
270 END SUB
```

Valentin's Method in HPGCC2:

```
#include <hpgcc49.h>
#include <tommath.h>

int main()
{
    mp_int f,x,y,z,a,m,t;
    char *buf;
    int i, j, n = 0, k, x1, start, g=0;

    sys_slowOff();
    clear_screen();
    printf("running...");
```

```
    x1 = sat_pop_zint_llong();
    k = sat_pop_zint_llong();

    mp_init(&f); mp_init(&x); mp_init(&y); mp_init(&z); mp_init(&a); mp_init(&m); mp_init(&t);
    start = sys_RTC_seconds();

    while(n+=10) {
        printf("%d ",n);
        mp_set_int(&f,1);
        for(i=2;i<=n;i++)
            mp_mul_d(&f,i,&f);              // f = n!
        mp_set_int(&y,x1);                  // y = x1
        for(i=2;i<=n;i++) {
            mp_set_int(&t,i-1);
            mp_div(&y,&t,&z,&a);            // z = floor(y/(i-1))
            mp_expt_d(&z,k,&x);             // x = z^k;
            mp_add(&x,&y,&z);               // z = x + y
            mp_div(&z,&f,&a,&y);            // y = z mod f
            mp_set_int(&t,i);
            mp_div(&y,&t,&a,&m);            // m = y mod i
            if(!mp_iszero(&m)) {
                buf = malloc((int)((log10(256.0)*mp_unsigned_bin_size(&m)) + 10) * sizeof(char));
                mp_toradix(&m, buf, 10);
                j = atoi(buf);
                printf("\n\nk:%d  x1:%d  index: %d\nremainder: %.12f\n",k,x1,i,(double)j/i);
                printf("f: %d\nx: %d\ny: %d\nz: %d\nm: %d\n",
                    (int)(log10(256.0)*mp_unsigned_bin_size(&f)),
                    (int)(log10(256.0)*mp_unsigned_bin_size(&x)),
                    (int)(log10(256.0)*mp_unsigned_bin_size(&y)),
                    (int)(log10(256.0)*mp_unsigned_bin_size(&z)),
                    (int)(log10(256.0)*mp_unsigned_bin_size(&m)));
                printf("time(s): %d",sys_RTC_seconds()-start);
                WAIT_CANCEL;
                return(0);
            }
        }
    }
}
```

Edited: 29 Apr 2008, 9:21 p.m.

Re: My Original Solutions and Comments - Part 3 (Re: S&SMC#20) [LONG and I mean it]

*Message #86 Posted by **Valentin Albillo** on 1 May 2008, 2:50 p.m.,*
in response to message #85 by Egan Ford

Hi, Egan:

> I'm truly wordless at your magnificent achievements with this and any other challenges you've taken on.
>
> As you know, I don't specially like RPL models and that's the reason why I have none (save for a NIB HP28S) and further I've never been interested in them or produced programs for them, but I must confess that the way you use them, what with HPGCC, is truly amazing and fully shows their incredible capabilities to the best.
>
> Just for instance, the mere fact you can solve the **k=3, x_1=11** case is frankly awesome in the extreme, as it demands 4300-digit arithmetic at the very least, yet everything's concluded within a mere half and hour in a handheld calculator.

Even sticking to the HP-71B, your non-fully multiprecision method is also outstanding, despite the fact that it doesn't solve all cases. The timing for that same case is also fantastic at 450+ seconds under Emu71.

In short, I've been as enthralled with your productions as you might have been with mine, if not more, and (said in good humor, mind you) you're the main culprit of my S&SMC series being ended: you've forced me to raise the level to such heights, in order to actually provide a credible challenge that could hold your attacks even for a few days or hours, that most anyone else couldn't follow at all and quickly lost interest finding them completely over the top !! :-)

Last, thank you very much for everything: your interest, your dedication, your insights, your amazing solutions. You've certainly been one of the main reasons for me enjoying it all and the best justification for my (considerable) efforts.

Best regards from V.

Re: My Original Solutions and Comments - Part 3 (Re: S&SMC#20) [LONG and I mean it]

*Message #87 Posted by **PeterP** on 30 Apr 2008, 12:35 a.m.,*
in response to message #84 by Valentin Albillo

Valentin,

Thanks for all the time you stole away from other endeavors of your live to bring yet another challenge to us. As usually I enjoyed it a great deal, learned a lot and even allow myself a tinge of happiness in having gotten farther in this challenge than in most of my other attempts (although your gracious allowance to my deviation from some rules was most appreciated).

And your *free* providing of a multi-precision library is a special goodie to end this sequence of challenges on a high note! I'll have to study the library carefully to understand it (maybe it even gives me some tips on how to write efficient multiplication and division routines in HP41 MCODE, something I am currently trying to learn and find info about)

Thanks again and I can only second Egans hope that around April 1st you will allow me again to make a fool of myself by attempting your 'simple' challenge.

Seriously, though, a very heartfelt Thank You! from your Austrian participant for all your kind work and advice you so freely give.

Cheers

Peter

Re: My Original Solutions and Comments - Part 3 (Re: S&SMC#20) [LONG and I mean it]

*Message #88 Posted by **Valentin Albillo** on 1 May 2008, 3:18 p.m.,*
in response to message #87 by PeterP

Hi, PeterP:

You've certainly done great things with this last challenge, I'm pretty amazed at your incresing capabilities over the months and the accelerated rate of prowess you've continually been showing. I've read your outputs with interest and admiration, and want to thank you for them and for taking the time to produce them.

As for my ad-hoc multiprecision library, it's *very* ad-hoc, concocted in as little time as possible and with the *minimum* functionality needed, as stated in the disclaimer. If you're gonna develop multiprecision routines in HP41C's MCODE, better starting code and algorithms would be needed. A minimum reasonable effort would consider these improvements as mandatory:

- A complete, symmetric set, which would include addition, subtraction, multiplication, and division, also including in-place versions and single-precision operands, plus modulus and powers

- Extra operations: fast multiplication/division by powers of ten (shift left and right), boolean comparisons (less than, greater than, equal, equal to zero)
- Provision to deal with negative arguments and, eventually, floating point arguments.
- Provision to exceed the 500-digit limit by not making it dependent on the built-in REAL range.
- Fully checked for implementation errors, and all argument or range errors fully trapped and duly reported in a controlled manner.

As you see, that's a lot of work, and implementing it all in MCODE can be a daunting task indeed, but with the potential to be very enjoyable and ultimately useful as well.

Thanks again, I'm keen to learn of your MCODE experiences, and

Best regards from V.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall