HP Forum Archive 15

Short & Sweet Math Challenge #14: Cooking Conjectures !

*Message #1 Posted by **Valentin Albillo** on 7 Mar 2006, 8:52 a.m.*

Hi all,

A month has elapsed, and March, 7th is a pretty good day for another S&SMC, this time **S&SMC#14, Cooking Conjectures !**

Preamble

Number Theory is a fascinating branch of Mathematics, where we mostly deal with such basic, fundamental entities as integer numbers. Despite the apparent simplicity, experience shows that it's actually very easy to make integer-related conjectures that seem extremely simple on the outside, yet are nearly *intractable*. A good example would be the infamous Fermat Last Theorem (FLT), which remained a conjecture for several centuries despite the strongest, most strenuous attempts at proving it.

FLT was at long last proved, thus leaving its *conjecture* status (despite its name) to actually become a *theorem* proper, but many other important conjectures are still awaiting for either some clever demonstration, which will break new barriers and advance knowledge, or else a counterexample which proves them false, which doesn't usually advance knowledge an inch but at least gives peace of mind and conclusively stops further costly, time-wasting attempts to try and prove them. When such a counterexample is found, we say that the conjecture is ***cooked***, i.e., unsound, false. For this very S&SMC#14, put on your chef's hat and let's try and cook some nice conjectures for dinner !

The Challenge

For each of these three *plausible* conjectures, you must find a *cook*, i.e., the lowest counterexample that falsifies them. In all cases you are done if you manage to find but *one* counterexample, within the stated ranges. Optimizing for maximum speed will be first priority, then for program size and simplicity. These are the conjectures to cook:

Conjecture 1: Well-done

Some famous mathematician of old noted the fact that, apparently, you could find any number of integer solutions to the equations:

$a^2 + b^2 = c^2$ (e.g.: $3^2 + 4^2 = 5^2$)
$a^3 + b^3 + c^3 = d^3$ (e.g.: $3^3 + 4^3 + 5^3 = 6^3$)
...

where the number of terms added up is the *same* as the power, but he was able to prove that there were no non-trivial integer solutions to:

$a^3 + b^3 = c^3$

where the number of terms added up was *less* than the power. He thus conjectured that such equations as:

$a^4 + b^4 + c^4 = d^4$
$a^5 + b^5 + c^5 + d^5 = e^5$
...

etc., would also have no non-trivial solutions at all. Now, *you must prove him wrong by finding a counterexample for the 5th-power case*, i.e., write a program that finds one solution of:

$\mathbf{a^5 + b^5 + c^5 + d^5 = e^5}$

for non-zero, positive integer values **a**, **b**, **c**, **d**, **e** less than 150. Such a solution eluded mathematicians for two centuries till it was found in 1966 using tremendous computing power for the time. Just duplicate the feat using your small HP handheld calculator and your programming ingenuity.

Conjecture 2: Medium

Another less well-known mathematician stated the following conjecture:

"Every positive integer greater than 5 is the sum of a *prime* and a *power*"

For instance:

...
$1234 = 991 + 3^5$
$1235 = 79 + 34^2$
$1236 = 11 + 35^2$
$1237 = 337 + 30^2$
$1238 = 13 + 35^2$
...

etc. *You must find the smallest counterexample N which cannot be expressed as the sum of a prime and a power*, limiting your search to the range from N = 6 to 2000. Of course the prime numbers are 2,3,5,... (so 1 is *not* considered to be a prime), and the powers are $1^1 = 1$, $2^2 = 4$, $2^3 = 8$, $3^2 = 9$, $2^4 = 16$, ..., (so 0 is *not* considered to be a power for the purposes of decomposition into the sum of a prime and a power, but 1 *is*).

Conjecture 3: Rare

Finally, a more recent and amusing conjecture is that all positive integer numbers can be made into a palindromic number (i.e., one which reads the same from right or from left, such as 123474321) by reversing their digits and adding the result to the original number, then repeating these steps until you get a palindromic number. For instance, 78 gets palindromic in 4 cycles:

```
cycle 1:    78 +   87 =  165
cycle 2:   165 +  561 =  726
cycle 3:   726 +  627 = 1353
cycle 4:  1353 + 3531 = 4884, palindromic
```

You must find the smallest alleged counterexample N, for N up to 200, which fails to produce a palindromic result after M cycles, where M should go up to 200 cycles minimum, preferably up to 1000 cycles. Any number N which fails to produce a palindromic result after M cycles, for suitably large M (say 200, 500, or 1000) will be considered a counterexample for the purposes of this challenge.

Your program must ask for the maximum number of cycles to perform, M, and must output *any and all values* of N up to 200 which do not produce a palindromic result. Be aware that the numbers involved will get *very big* very soon. Your program must cater for this, without ever producing overflow or losing significant digits.

Caveat Emptor

The usual caveats apply, namely:

1. Do *not* post just solutions, actual code is *mandatory*. If you won't post code to accompany your alleged solutions at the time of posting them, then do not post the solutions, period. Wait till you can post both.

2. Code must be for an HP calculator preferably, but other vintage handhelds are acceptable as well. Posting code written in Visual Basic, Java, FORTRAN, or any other PC language will be considered unpolite and disrespectful.

3. Try to achieve a proper balance between you manually doing most of the work and having a "PRINT (solution)" program which does nothing of interest, and you contributing no enlightening ideas of your own and having instead a dumb, brute-force program which delivers the goods but takes ages to run. The ideal is to use some clever ingenuity to significantly speed up the program while still letting it do the hard work as it should. Remember the old computerese proverb:

"The Program was made for man, not man for the Program"

Anyway, you'll need some reasonable algorithmic and programming ingenuity if you don't want to kill your batteries in one go.

I'll post my own original solutions next Monday, which will be *three simple programs 7-, 8-, and 9-lines long for the HP-71B*. Though far from being the last word in state-of-the-art programming, they're didactically simple and they're fast and they get the work done reasonably quickly. I'll post background and relevant comments as well.

Let's see yours.

Best regards from V.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #2 Posted by ***GE*** *on 8 Mar 2006, 9:45 a.m.,*
in response to message #1 by Valentin Albillo

</lurk mode> Regarding challenge 3, I "know" that the answer is 196. <lurk mode> very much appreciating your challenges (without ever submitting code...)

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #3 Posted by ***Bram*** *on 9 Mar 2006, 6:35 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

Nice challenges and nicely formulated. I immediately started thinking about them, but, unpleasantly enough, I can't think of any other solution than to scan possibilities, which is disapproved in the first place. So for the moment no contributions from my side.
Still, as I was anxious to know the answer, I did write a brute-force program for my 32SII for the first challenge to see how fast it would run. Well, fast enough, but the amount of computations will take a few days to complete, despite the fact that I don't examine combinations that I already have computed but in different order.
Obviously this #14 is not my cup of tea, but I like reading it and I'm looking forward to your approaches. As always.

groeten,
Bram

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #4 Posted by ***Valentin Albillo*** *on 9 Mar 2006, 7:24 a.m.,* *in response to message #3 by Bram*

Hi, Bram:

Bram posted:

"Obviously this #14 is not my cup of tea, but I like reading it and I'm looking forward to your approaches. As always."

> Thanks for your interest and kind words but don't despair so soon. Even if Conjecture 1 would seem hard, it actually isn't that much, and you still have Conjectures 2 & 3 which are easier to cook.
>
> However, the HP32SII is probably not the ideal calc for this #14, because of its extremely limited amount of RAM. After you've entered some clever programming you'll be left with too few bytes for necessary variables, and even the program itself can't be much more refined than a pure brute-force search in that little RAM.
>
> If you can, I suggest you try instead one of these models: HP41CV/CX, HP42S, HP-71B, HP48/49 series. Even if you don't have any of these models available, you can always get Emu71 or other free emulators from the net, and try your might with that.
>
> Don't give up, I know you can succeed ! :-) You might also consider that if all interested people simply get the lazy bug and decide to just wait and see whatever solutions I will eventually post, I might consider the response a total failure and stop altogether posting any further challenges ... in other words, if I do work hard to produce them in the first place, then interested people are expected to work as well trying to solve them for good, else no deal.
>
> This isn't intended as a showcase of my abilities but an interaction between all of us who care for this, where we'll get to see some fun math and interesting programming techniques for a variety of models, and hopefully enjoy it all and learn something valuable in the process.

Best regards from V.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

*Message #5 Posted by **Maximilian Hohmann** on 9 Mar 2006, 7:53 a.m.,*
in response to message #3 by Bram

Hello!

Thank you for this nice challenge, really something worth thinking about whike driving to work and back...

> Quote:

> Bram: Nice challenges and nicely formulated. I immediately started thinking about them, but, unpleasantly enough, I can't think of any other solution than to scan possibilities, which is disapproved in the first place. So for the moment no contributions from my side.

The same for me, I'm afraid. Nevertheless (I'm an engineer, not a mathematician ;-)), I tried a brute force "attack" on challenge no.1 with a little C-programme on my PowerBook (I wish I had this Casio pocket calculator with "C" programming language, that would make my attempt a valid entry...) and it took 18 seconds (*) to find the result(s) (**). Which means: definitely no answer within my remaining lifetime on any vintage programmable calculator!

The only alternatives that come to mind would be an iterative approach or an optimisation (simplex method?) approach, but both would not fit into any of my programmable calculators.
Looking forward to see the real smart solutions!

Greetings, Max

(*) and 27 Minutes, if searching up to 300 instead of 150 ... which probably tranlates to a millenium on the hp-41
(**) my very first real brute-brute-brute approach would have taken several hours, so at least *some* thinking had to go into it.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #6 Posted by ***Valentin Albillo*** *on 9 Mar 2006, 8:29 a.m.,*
in response to message #5 by Maximilian Hohmann

Hi, Maximilian:

Maximilian posted:

"Thank you for this nice challenge, really something worth thinking about whike driving to work and back... "

> You're welcome, glad you find it interesting.

"I tried a brute force "attack" on challenge no.1 [...] and it took 18 seconds [...] which means: definitely no answer within my remaining lifetime on any vintage programmable calculator!"

> Believe me, it's not as hard as it seems. Obviously launching a pure brute-force search with the equivalent of four or five for-next loops is certain to take ages, as I mentioned in my original posting. But some clever refinements plus careful reading of the given conditions can make all the difference in the world, to the point where a vintage handheld, and certainly such models as the modern 48/49 series, can solve it in a few hours at most, if not mere minutes.

"Looking forward to see the real smart solutions! "

> I think you're overestimating the real difficulty. That's usually the way with my challenges: most people find them extremely difficult at first, till they realize that they're actually quite manageable, even easy given the right approach. Else, I wouldn't post them, it's no fun to ask solutions to challenges that would forcibly require the use of a fast, full-fledged computer.

Go ahead and if you don't manage to cope with Conjecture 1, try Conjectures 2 & 3 instead, they're probably easier (that's why I labelled them "Well-done" (hardest), "Medimum" (so-so hard) and "Rare" (easiest) :-)

Thanks for your interest and best regards from V.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #7 Posted by **.** *on 9 Mar 2006, 8:34 a.m.,*
in response to message #5 by Maximilian Hohmann

Hi,

Any chance that you could put your C solution online? I'd be happy to port it to the HP49 series and make it a valid entry (yes, you can use C with these, and the result is *very* fast).

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #8 Posted by ***Maximilian Hohmann*** *on 9 Mar 2006, 8:43 a.m.,*
in response to message #7 by .

Hi!

> Quote:
>
> Any chance that you could put your C solution online? I'd be happy to port it to the HP49 series and make it a valid entry (yes, you can use C with these, and the result is *very* fast).

I have uploaded it here: http://www.bombie.de/tmp/hochfuenf.c

Greetings, Max

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #9 Posted by ***Marcus von Cube, Germany*** *on 9 Mar 2006, 11:42 a.m.,*
in response to message #8 by Maximilian Hohmann

Hi Max,

a PB-2000C version is in your mailbox.

Marcus

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #10 Posted by ***Marcus von Cube, Germany*** *on 11 Mar 2006, 9:05 a.m.,*
in response to message #5 by Maximilian Hohmann

Hi Max & Valentin,

> Quote:
>
> (I wish I had this Casio pocket calculator with "C" programming language, that would make my attempt a valid entry...) and it took 18 seconds (*) to find the result(s) (**)

I *do* have a PB-2000C and I was able to drain a set of expensive batteries with the program without ever coming near a solution...

So I tried to do something about it and I shortened the loops as good as I could. Here is the program:

```
/*
  Short&Sweet Math Challenges No.14 http://www.hpmuseum.org/


  Version for Casio PB-2000C


*/


/* IMIN: starting point */
#define IMIN 150
```

```
/* IMAX: highest number to search */
#define IMAX 150


/* There is no command line in Casio C */
main()
{
  double a5     = 0.0;
  double b5     = 0.0;
  double c5     = 0.0;
  double d5     = 0.0;
  double e5     = 0.0;
  double sum    = 0.0;
  double sumc   = 0.0;
  double sumd   = 0.0;
  double diff   = 0.0;
  double z      = 0.0;
  double vsmall = 1.0E-10;
  int    imin   = 0;
  int    imax   = 0;
  int    i      = 0;
  int    ia     = 0;
  int    ib     = 0;
  int    ic     = 0;
  int    id     = 0;
  int    ie     = 0;
  double *powtab = NULL;
  int    lpowtab = 0;
  int    output  = 0;


  /* get the parameters */
  printf( "imin=" );
  scanf( "%d", &imin );
  if ( imin == 0 ) imin = IMIN;


  printf( "imax=" );
  scanf( "%d", &imax );
  if ( imax == 0 ) imax = IMAX;


  printf( "output=" );
  scanf( "%d", &output );


  printf("SSMC #14.1: Searching up to %d\n", imax);
```

```
/* power table to save multiplictions */
lpowtab = imax + 1;
powtab  = (double *) malloc(lpowtab * sizeof(powtab[0]));


if (powtab == NULL) {
  printf("*** ERROR ***\nMemory allocation failed.\n");
  return;
}


for (i = 1 ; i < imin ; i++) {
  z = (double) i;
  powtab[i] = z * z * z * z * z;
}


/* outermost loop for "e" */
for (ie = imin ; ie <= imax ; ie++) {
  z = (double) ie;
  e5 = powtab[ie] = z * z * z * z * z;
  if (output) {
    clrscr();
  }
  printf("e=%-4d e^5=%-15.0f\n", ie, e5);


  /* compute lower bound of "d" loop */
  z = e5 - 3 * powtab[ie-1];
  id = z <= 2 ? 1 : (int) pow( z, 0.2 );
  if (output) {
    gotoxy(12,1);
    printf("%-3d",id);
  }
  for (; id < ie; id++) {
    d5 = powtab[id];
    if (output) {
      gotoxy(12,1);
      printf("%-3d",id);
      gotoxy(16,1);
      printf("%-15.0f",d5);
    }


    /* compute lower bound of "c" loop */
    z = e5 - 3 * d5;
    ic = z <= 2 ? 1 : (int) pow( z, 0.2 );
```

```
    if (output) {
      gotoxy(0,1);
      printf("        %-3d",ic);
    }
    for (; ic <= id; ic++) {
      c5 = powtab[ic];
      sumc = d5 + c5;
      if (output) {
        gotoxy(8,1);
        printf("%-3d",ic);
        gotoxy(16,1);
        printf("%-15.0f",sumc);
      }
      /* check if c^5 + d^5 exeeds e^5 */
      if ( sumc > e5 ) break;


      /* compute lower bound of "b" loop */
      z = e5 - d5 - 2 * c5;
      ib = z <= 2 ? 1 : (int) pow( z, 0.2 );
      if (output) {
        gotoxy(0,1);
        printf("    %-3d",ib);
      }


      for (; ib <= ic; ib++) {
        b5 = powtab[ib];
        sumd = sumc + b5;
        if (output) {
          gotoxy(4,1);
          printf("%-3d",ib);
          gotoxy(16,1);
          printf("%-15.0f",sumd);
        }
        /* check if b^5 + c^5 + d^5 exeeds e^5 */
        if ( sumd > e5 ) break;


        /* compute lower bound of "a" loop */
        z = e5 - sumd;
        ia = z <= 2 ? 1 : (int) pow( z, 0.2 );
        if (output) {
          gotoxy(0,1);
          printf("%-3d",ia);
        }
        for (; ia <= ib; ia++) {
          a5 = powtab[ia];
```

```
          sum = sumd + a5;
          diff = sum - e5;


          if (-vsmall < diff && diff < vsmall) {
            if (output) clrscr();
            printf("SOLUTION: a=%4d b=%4d c=%4d\n          d=%4d e=%4d\n",
                   ia, ib, ic, id, ie);
            beep(1);
            if (output) getchar();
          }
          else if ( sum > e5 ) {
            /* a^5 + b^5 + c^5 + d^5 exeeds e^5 */
            break;
          }
          else {
            if (output) {
              gotoxy(0,1);
              printf("%-3d",ia);
              gotoxy(16,1);
              printf("%-15.0f",sum);
            }
          }
        }
      }
    }
  }
}


free(powtab);
powtab = NULL;


printf("\nFinished\n");
}
```

I'm relucant to post the result, because I've only tested the program above with time consuming screen output enabled and even after giving it a whole night it didn't reach more than e=70. At the time of this writing, the software is running again, this time with a restricted set of values for e to check ;-). I hope that it will come to a solution before Sunday.

To provide a solution for an HP calculator of old, I ported it back to DOS Borland C and let it run on my venerable HP 200lx. The machine seems to be much faster than the Casio, having reached e=65 after about 15 or 20 minutes. The Casio runs interpreted code (just like BASIC) while the HP runs a compiled DOS program.

I strongly believe that the search algorithm must be improved greatly to have a chance on a slow calculator.

Marcus

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #11 Posted by ***Marcus von Cube, Germany*** *on 11 Mar 2006, 6:25 p.m.,*
in response to message #10 by Marcus von Cube, Germany

After a few hours of hard computational work, the HP 200lx was finally able to deliver:

27^5+84^5+110^5+133^5=144^5

My poor Casio has at least arrived at showing the same result after being told to start with e=144...

Re: S&SMC#14 Part 3 - BASIC solution

Message #12 Posted by ***Marcus von Cube, Germany*** *on 9 Mar 2006, 8:23 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

this time I had some spare time to try to solve at least one of your challenges. Here is a BASIC program for the third conjecture.

The program

```
(Indentation and empty lines added later in the listing)


100 REM S&SMC 14, Problem 3
110 INPUT "Number of cycles=";NC
120 INPUT "From=";N1,"To=";N2
130 DIM D(1,199):REM digits


140 FOR N=N1 TO N2
160  REM Split N in digits
170  ID=0:L=0
180  T=N
190  D(ID,L)=T-10*INT(T/10)
200  T=INT(T/10)
```

```
210  L=L+1
220  IF T<>0 THEN 190
225  L=L-1


230  REM Cycle loop
240  FOR C=1 TO NC
250   GOSUB 1000:REM display
260   REM Add and Test
270   F=-1:CY=0


280   FOR I=0 TO L
290    D1=D(ID,I):D2=D(ID,L-I)
300    F=F AND (D1=D2)
310    IF F AND (I>=L-I)THEN I=L:C=NC:GOTO 350:REM Palindromic
320    S=D1+D2+CY:CY=0
330    IF S>9 THEN S=S-10:CY=1
340    D(1-ID,I)=S
350   NEXT I


360   ID=1-ID:REM swap source&dest
365   IF CY=1 THEN L=L+1:D(ID,L)=1
370  NEXT C


380  IF F=0 THEN 410
390 NEXT N
400 END


1000 REM display
1005 PRINT N;C;":";
1010 FOR II=0 TO L
1020  PRINT CHR$(48+D(ID,II));
1030 NEXT II
1040 PRINT
1050 RETURN
```

I wrote the program on a TI-74 with bare 8K RAM. The BASIC used here is pretty standard and I assume that a HP-71 could run the same software with minor changes (: -> @).

How is it done?

I didn't make the attempt to handle the sums as integers or real numbers because the number of digits available (10 to 14) wouldn't be enough. Splitting an array of 10 digit numbers in individual digits seemed to be too time consuming. So I traded memory for speed in this first attempt. Each individual digit is stored as a number in an array.

The array is defined with DIM D(1,199). TI-Basic does not support single precision or integer variables so this is a real waste of memory. The array caters for two numbers with up to 200 digits. A posible modification could be to use two character strings (with a maximum length of 255 in most BASICs) or to PEEK/POKE directly into memory.

The first dimension is used to switch between the source and the destination number. There is only one source number, read from left to right for the first summand and from right to left for the second. The other number is the sum. After each cycle, the roles are changed (see line 360).

While adding, two things happen. A comparison is made between the two digits added (line 300) and the result of the comparison is ANDed into a flag (F). If the flag is still set after half of the digits are processed, the number is a palindrome and the rest of the two inner loops are skipped (line 310.)

Secondly, a carry (CY) is set whenever the sum is greater than 9. This carry is than added in the next step of the digit loop (lines 320-340.)

The results

The program took about half an hour to find the result 196 with 200 cycles. The size of the array should be enough for more than 400 cycles (the longest number I got has 88 digits). The following output was created with a modified version of the software that can PRINT to a datafile on the PC (via the PC interface).

```
1  1 :1
2  1 :2
3  1 :3
4  1 :4
5  1 :5
6  1 :6
7  1 :7
8  1 :8
9  1 :9
10  1 :10
10  2 :11
11  1 :11
12  1 :12
12  2 :33
13  1 :13
13  2 :44
14  1 :14
14  2 :55
15  1 :15
15  2 :66
```

```
16  1 :16
16  2 :77
17  1 :17
17  2 :88
18  1 :18
18  2 :99
19  1 :19
19  2 :110
19  3 :121
20  1 :20
20  2 :22
21  1 :21
21  2 :33
22  1 :22
23  1 :23
23  2 :55
24  1 :24
24  2 :66
25  1 :25
25  2 :77
26  1 :26
26  2 :88
27  1 :27
27  2 :99
28  1 :28
28  2 :110
28  3 :121
29  1 :29
29  2 :121
30  1 :30
30  2 :33
31  1 :31
31  2 :44
32  1 :32
32  2 :55
33  1 :33
34  1 :34
34  2 :77
35  1 :35
35  2 :88
36  1 :36
36  2 :99
37  1 :37
37  2 :110
37  3 :121
38  1 :38
38  2 :121
```

```
39  1 :39
39  2 :132
39  3 :363
40  1 :40
40  2 :44
41  1 :41
41  2 :55
42  1 :42
42  2 :66
43  1 :43
43  2 :77
44  1 :44
45  1 :45
45  2 :99
46  1 :46
46  2 :110
46  3 :121
47  1 :47
47  2 :121
48  1 :48
48  2 :132
48  3 :363
49  1 :49
49  2 :143
49  3 :484
50  1 :50
50  2 :55
51  1 :51
51  2 :66
52  1 :52
52  2 :77
53  1 :53
53  2 :88
54  1 :54
54  2 :99
55  1 :55
56  1 :56
56  2 :121
57  1 :57
57  2 :132
57  3 :363
58  1 :58
58  2 :143
58  3 :484
59  1 :59
59  2 :154
59  3 :605
```

```
59  4 :1111
60  1 :60
60  2 :66
61  1 :61
61  2 :77
62  1 :62
62  2 :88
63  1 :63
63  2 :99
64  1 :64
64  2 :110
64  3 :121
65  1 :65
65  2 :121
66  1 :66
67  1 :67
67  2 :143
67  3 :484
68  1 :68
68  2 :154
68  3 :605
68  4 :1111
69  1 :69
69  2 :165
69  3 :726
69  4 :1353
69  5 :4884
70  1 :70
70  2 :77
71  1 :71
71  2 :88
72  1 :72
72  2 :99
73  1 :73
73  2 :110
73  3 :121
74  1 :74
74  2 :121
75  1 :75
75  2 :132
75  3 :363
76  1 :76
76  2 :143
76  3 :484
77  1 :77
78  1 :78
78  2 :165
```

```
78  3 :726
78  4 :1353
78  5 :4884
79  1 :79
79  2 :176
79  3 :847
79  4 :1595
79  5 :7546
79  6 :14003
79  7 :44044
80  1 :80
80  2 :88
81  1 :81
81  2 :99
82  1 :82
82  2 :110
82  3 :121
83  1 :83
83  2 :121
84  1 :84
84  2 :132
84  3 :363
85  1 :85
85  2 :143
85  3 :484
86  1 :86
86  2 :154
86  3 :605
86  4 :1111
87  1 :87
87  2 :165
87  3 :726
87  4 :1353
87  5 :4884
88  1 :88
89  1 :89
89  2 :187
89  3 :968
89  4 :1837
89  5 :9218
89  6 :17347
89  7 :91718
89  8 :173437
89  9 :907808
89  10 :1716517
89  11 :8872688
89  12 :17735476
```

```
89  13 :85189247
89  14 :159487405
89  15 :664272356
89  16 :1317544822
89  17 :3602001953
89  18 :7193004016
89  19 :13297007933
89  20 :47267087164
89  21 :93445163438
89  22 :176881317877
89  23 :955594506548
89  24 :1801200002107
89  25 :8813200023188
90  1 :90
90  2 :99
91  1 :91
91  2 :110
91  3 :121
92  1 :92
92  2 :121
93  1 :93
93  2 :132
93  3 :363
94  1 :94
94  2 :143
94  3 :484
95  1 :95
95  2 :154
95  3 :605
95  4 :1111
96  1 :96
96  2 :165
96  3 :726
96  4 :1353
96  5 :4884
97  1 :97
97  2 :176
97  3 :847
97  4 :1595
97  5 :7546
97  6 :14003
97  7 :44044
98  1 :98
98  2 :187
98  3 :968
98  4 :1837
98  5 :9218
```

```
98  6 :17347
98  7 :91718
98  8 :173437
98  9 :907808
98  10 :1716517
98  11 :8872688
98  12 :17735476
98  13 :85189247
98  14 :159487405
98  15 :664272356
98  16 :1317544822
98  17 :3602001953
98  18 :7193004016
98  19 :13297007933
98  20 :47267087164
98  21 :93445163438
98  22 :176881317877
98  23 :955594506548
98  24 :1801200002107
98  25 :8813200023188
99  1 :99
100  1 :100
100  2 :101
101  1 :101
102  1 :102
102  2 :303
103  1 :103
103  2 :404
104  1 :104
104  2 :505
105  1 :105
105  2 :606
106  1 :106
106  2 :707
107  1 :107
107  2 :808
108  1 :108
108  2 :909
109  1 :109
109  2 :1010
109  3 :1111
110  1 :110
110  2 :121
111  1 :111
112  1 :112
112  2 :323
113  1 :113
```

```
113  2 :424
114  1 :114
114  2 :525
115  1 :115
115  2 :626
116  1 :116
116  2 :727
117  1 :117
117  2 :828
118  1 :118
118  2 :929
119  1 :119
119  2 :1030
119  3 :1331
120  1 :120
120  2 :141
121  1 :121
122  1 :122
122  2 :343
123  1 :123
123  2 :444
124  1 :124
124  2 :545
125  1 :125
125  2 :646
126  1 :126
126  2 :747
127  1 :127
127  2 :848
128  1 :128
128  2 :949
129  1 :129
129  2 :1050
129  3 :1551
130  1 :130
130  2 :161
131  1 :131
132  1 :132
132  2 :363
133  1 :133
133  2 :464
134  1 :134
134  2 :565
135  1 :135
135  2 :666
136  1 :136
136  2 :767
```

```
137  1 :137
137  2 :868
138  1 :138
138  2 :969
139  1 :139
139  2 :1070
139  3 :1771
140  1 :140
140  2 :181
141  1 :141
142  1 :142
142  2 :383
143  1 :143
143  2 :484
144  1 :144
144  2 :585
145  1 :145
145  2 :686
146  1 :146
146  2 :787
147  1 :147
147  2 :888
148  1 :148
148  2 :989
149  1 :149
149  2 :1090
149  3 :1991
150  1 :150
150  2 :201
150  3 :303
151  1 :151
152  1 :152
152  2 :403
152  3 :707
153  1 :153
153  2 :504
153  3 :909
154  1 :154
154  2 :605
154  3 :1111
155  1 :155
155  2 :706
155  3 :1313
155  4 :4444
156  1 :156
156  2 :807
156  3 :1515
```

```
156  4 :6666
157  1 :157
157  2 :908
157  3 :1717
157  4 :8888
158  1 :158
158  2 :1009
158  3 :10010
158  4 :11011
159  1 :159
159  2 :1110
159  3 :1221
160  1 :160
160  2 :221
160  3 :343
161  1 :161
162  1 :162
162  2 :423
162  3 :747
163  1 :163
163  2 :524
163  3 :949
164  1 :164
164  2 :625
164  3 :1151
164  4 :2662
165  1 :165
165  2 :726
165  3 :1353
165  4 :4884
166  1 :166
166  2 :827
166  3 :1555
166  4 :7106
166  5 :13123
166  6 :45254
167  1 :167
167  2 :928
167  3 :1757
167  4 :9328
167  5 :17567
167  6 :94138
167  7 :177287
167  8 :960058
167  9 :1810127
167  10 :9020308
167  11 :17050517
```

```
167  12 :88555588
168  1 :168
168  2 :1029
168  3 :10230
168  4 :13431
169  1 :169
169  2 :1130
169  3 :1441
170  1 :170
170  2 :241
170  3 :383
171  1 :171
172  1 :172
172  2 :443
172  3 :787
173  1 :173
173  2 :544
173  3 :989
174  1 :174
174  2 :645
174  3 :1191
174  4 :3102
174  5 :5115
175  1 :175
175  2 :746
175  3 :1393
175  4 :5324
175  5 :9559
176  1 :176
176  2 :847
176  3 :1595
176  4 :7546
176  5 :14003
176  6 :44044
177  1 :177
177  2 :948
177  3 :1797
177  4 :9768
177  5 :18447
177  6 :92928
177  7 :175857
177  8 :934428
177  9 :1758867
177  10 :9447438
177  11 :17794887
177  12 :96644658
177  13 :182289327
```

```
177  14 :906271608
177  15 :1712444217
177  16 :8836886388
178  1 :178
178  2 :1049
178  3 :10450
178  4 :15851
179  1 :179
179  2 :1150
179  3 :1661
180  1 :180
180  2 :261
180  3 :423
180  4 :747
181  1 :181
182  1 :182
182  2 :463
182  3 :827
182  4 :1555
182  5 :7106
182  6 :13123
182  7 :45254
183  1 :183
183  2 :564
183  3 :1029
183  4 :10230
183  5 :13431
184  1 :184
184  2 :665
184  3 :1231
184  4 :2552
185  1 :185
185  2 :766
185  3 :1433
185  4 :4774
186  1 :186
186  2 :867
186  3 :1635
186  4 :6996
187  1 :187
187  2 :968
187  3 :1837
187  4 :9218
187  5 :17347
187  6 :91718
187  7 :173437
187  8 :907808
```

```
187  9 :1716517
187  10 :8872688
187  11 :17735476
187  12 :85189247
187  13 :159487405
187  14 :664272356
187  15 :1317544822
187  16 :3602001953
187  17 :7193004016
187  18 :13297007933
187  19 :47267087164
187  20 :93445163438
187  21 :176881317877
187  22 :955594506548
187  23 :1801200002107
187  24 :8813200023188
188  1 :188
188  2 :1069
188  3 :10670
188  4 :18271
188  5 :35552
188  6 :61105
188  7 :111221
188  8 :233332
189  1 :189
189  2 :1170
189  3 :1881
190  1 :190
190  2 :281
190  3 :463
190  4 :827
190  5 :1555
190  6 :7106
190  7 :13123
190  8 :45254
191  1 :191
192  1 :192
192  2 :483
192  3 :867
192  4 :1635
192  5 :6996
193  1 :193
193  2 :584
193  3 :1069
193  4 :10670
193  5 :18271
193  6 :35552
```

```
193  7 :61105
193  8 :111221
193  9 :233332
194  1 :194
194  2 :685
194  3 :1271
194  4 :2992
195  1 :195
195  2 :786
195  3 :1473
195  4 :5214
195  5 :9339
196  1 :196
196  2 :887
196  3 :1675
196  4 :7436
196  5 :13783
196  6 :52514
196  7 :94039
196  8 :187088
196  9 :1067869
196  10 :10755470
196  11 :18211171
196  12 :35322452
196  13 :60744805
196  14 :111589511
196  15 :227574622
196  16 :454050344
196  17 :897100798
196  18 :1794102596
196  19 :8746117567
196  20 :16403234045
196  21 :70446464506
196  22 :130992928913
196  23 :450822227944
196  24 :900544455998
196  25 :1800098901007
196  26 :8801197801088
196  27 :17602285712176
196  28 :84724043932847
196  29 :159547977975595
196  30 :755127757721546
196  31 :1400255515443103
196  32 :4413700670963144
196  33 :8827391431036288
196  34 :17653692772973576
196  35 :85191620502609247
```

```
196  36 :159482241005228405
196  37 :664304741147513356
196  38 :1317620482294916822
196  39 :3603815405135183953
196  40 :7197630720180367016
196  41 :13305261530450734933
196  42 :47248966933966985264
196  43 :93507933867933969538
196  44 :177104867844767940077
196  45 :947154635293536341848
196  46 :1795298270686072793597
196  47 :9749270977546801719568
196  48 :18408442064004592449047
196  49 :92502871604050616929528
196  50 :175095833209091234750057
196  51 :925153265399993573340628
196  52 :1751196640799987135692157
196  53 :9264161958699957602603728
196  54 :17537224026299926194218357
196  55 :92918473189299188236491928
196  56 :175837936477498486373973857
196  57 :934217310162393261013712428
196  58 :1758434620324786522027424867
196  59 :9442681822581660752291773438
196  60 :17786453745152322604573635887
196  61 :96640091285774647759309104658
196  62 :182280281681549295517528109327
196  63 :906182107397142240703710191608
196  64 :1712373124704184482497411473217
196  65 :8836114272647029296571625205388
196  66 :17671139534403958504034349321776
196  67 :85383533877444544434477942439447
196  68 :159876958854887988978955775977805
196  69 :668656536414767878767414635656756
196  70 :1326313072829535757534829271313622
196  71 :3589444802113893332894111974449853
196  72 :7178889593228875666877224058899706
196  73 :13258878097456662332665448018788423
196  74 :45747659181913285659330927106673654
196  75 :91385319354816681317562845302348408
196  76 :171869639709643252636224690693706727
196  77 :899477035806065888888571598630674898
196  78 :1797953072701241777777132207161449896
196  79 :8787394689723559555548553279865047867
196  80 :16474800379447118011108106559729985745
196  81 :71233793175007298122189281057030833206
196  82 :131467596250025596244378551114170566423
```

```
196  83 :456132667661181469687074071166866330554
196  84 :911166336322351940474038252333632562208
196  85 :1713431572655604770948087405557266223327
196  86 :8946658200210652579438861471120017566498
196  87 :17893315300422394267788614031240046132996
196  88 :87816479304635435956564863353640397472867
196  89 :164643958609270772803130816807280794934745
196  90 :712083455691979390834439093880187654281206
196  91 :1314265912473067781768877187859384208661423
196  92 :4555933937312655599557549065463126404285554
196  93 :9111757983526301209015109021025263797681108
196  94 :17123625957151502418030218042061517695252227
196  95 :89348885628667526499233299462576693647884398
196  96 :178697760268335052998466598925153376306768796
196  97 :876565363941686582894131498175687238374565667
196  98 :1643130837774473154788262996461373387738131345
196  99 :7074449215608204801780891870975118165118444806
196  100 :13158897331226320592562872742059146230247889513
196  101 :44757771534490515617290699271561508443627774644
196  102 :89405544168971032134590308543213017887145550388
196  103 :177711098347842063369170618086336035873290100886
196  104 :865712190726372697049986690049696284617180218657
196  105 :1622524272442855393990083379990492558244271436225
196  106 :6848865996871408334989817180984428140686995688486
196  107 :13697731993731826579880634370878766182473991376972
196  108 :41665051931159993367687977979776328996213905156603
196  109 :72330202862429975735485955958452668991327820213217
196  110 :143561405734749962360971911916906426983754640416544
196  111 :589175452192139586970591031095969696931192144581885
196  112 :1177360893483279283940181161291049382862483399153870
196  113 :1960880827325962123342102773101543212586327379791581
196  114 :3812860564562814246793116545113976425281564660672272
196  115 :6535621229214639493586232001227952849464219311354455
196  116 :12080152368339288976183453003554806798828348532619811
196  117 :23971775952722178737028983038992974787121734857727832
196  118 :47844551796434357484958966077975048574244460815445764
196  119 :94599003602878605069016943144960997049587930530990638
196  120 :178208907106857199148923887279922093100275751161090187
196  121 :959299068264429200539153860068251935092034352870893058
196  122 :1809697146517859491078306720136603870094958815731886017
196  123 :8916578521706454391861373030412642572044545972149855098
196  124 :17822167934501908794613835170716374253979092043408611296
196  125 :87033848368531006729861196877870205903760002587384734167
196  126 :163177596747051013460811404755739322796520016173769567245
196  127 :705943564118661039158035342313143440860830166921465338606
196  128 :1312777128248322077226079683626386971712760333732930688113
196  129 :4431637520621652749397876519890256677940462572161148460244
```

```
196  130 :8852285932234405389895643040879413465879935133421405821588
196  131 :17703570973477720789681286190659816931869770177743801644176
196  132 :84848181808254828586495248086268985150568472955181709174947
196  133 :159795372526410756073000407072537069410037055800462527359795
196  134 :757749097790419306803015367807807773410407712815087800957746
196  135 :1405508106570937524507029745516516536920716316729185591905503
196  136 :4460600062390213660677326101672672016127770574119941609960544
196  137 :8911299123889328411454542204435433032365531237240874210021188
196  138 :17722499248669755732810174507780777054820072385480757429942386
196  139 :86047491724378214059813019585489547601921896141277441729365157
196  140 :161203884438855428229625930260088006192953791182564784448839225
196  141 :684142728926320709426985221860968068232480714007123618937141386
196  142 :1367284468742642409844069454721837136355070338914147248764382872
196  143 :4150119147170056608174774991039218410904674827956609727409210503
196  144 :7200248194449123205459539081187347712899449546023110444828321017
196  145 :14301486478889236411918989063364785524708809091046329889746741044
196  146 :58316251277781600431009869805923531860807791002509628777215151385
196  147 :116631402555564290951029640612737064811704681015910247554430412770
196  148 :193845437011306310461216047731197802027751601175002713109634549381
196  149 :377790873912623511032322205451406593165492213339016316220369097772
196  150 :655581836935237121965634500012802197319994436569131642439747195545
196  151 :1201173584869483253931268999926593405529999873138253374979385381101
196  152 :2213009424664216782245058999181637361829998494531777224664239092122
196  153 :4425918749328444553600007998463274723649996999954653349328488095244
196  154 :8851827597567878118200004997926549447298994000018207797567966290488
196  155 :17692754295225855146300009996853998903596988000046326585225923572076
196  156 :84720287247484417510300098966384988839466978000410482437485169301747
196  157 :159430683405957845911700186932878877787833967000712053885959447504495
196  158 :753836428365546196128700956271666656666073648007831602645463833539446
196  159 :1398771766730092402267401802642333313332246307015653294291027658177803
196  160 :4486490333932017325832508839064666446664708388063275337191404329956734
196  161 :8863089567973934661556117677139332893329317776115660574293797660903578
196  162 :17616180235947859412221234454278566875668635543232212238687595320707266
196  163 :83886882595526542633444469007965224742255880986444433734562548528868937
196  164 :157873765180053086366888937916820449484512851082888867359125108057737775
196  165 :735611515981575040135777218075035934428541470822777531039475189625116526
196  166 :1361223042963149970271554446149181758868072041635555062080050379140233063
196  167 :4964543462693650772877109807551890447439891458080106782879464071543454694
196  168 :9929086914398300555753120616093879794880873015169124565650028034186909388
196  169 :18768183728706501121407340231197660679860656921329338141200066968383718687
196  170 :97449922115672501335590732544163267577467336034533708553310627751121905468
196  171 :183900834231445102671171466087226644055043572179057418106621155402244899947
196  172 :933899276435996229272986217058501984605490194959721589282822699534682909328
196  173 :1757808562871992457555971345017993079111979300810434278555745399069355907667
196  174 :9424904102481927933114695685198032870231683297915866074113288390852013995238
196  175 :17750897205062866756229402370395956731552465606831732038226685682694028089487
196  176 :96248979254691525418512426084256613157066231566139052530492452508744307895258
```

```
196  177 :181508849599472050847915951177423126423141363231387114952073905028389605879527
196  178 :907487356583292560218175362960555489564465984556158274471821955303385554684708
196  179 :1714973812166596119346349835812210979128931969111227538043634020595771208469417
196  180 :8864621833942546323709758193033330670527151759233412927480073137552383392263588
196  181 :17718244767775103637410605485176660242044402519566716846059146374004876773528276
196  182 :85000782535615151001605670346943251762488426726233875296660620004162653517810047
196  183 :159002654071241291004212339604776514524976853441468839604311230019314307046510105
196  184 :660018294774655201036325746543640658883656278857146246537523630211456477502711056
196  185 :1310135500549309313072651482186282417756312667713192592185047260314012954995521122
196  186 :3521391495141413443700057295139195595418449244856019405026609963453052405050831253
196  187 :7042772000183916987399123500188302179847897390811938720954110036896193820992762506
196  188 :13095444990467833973699238090466693360785884880523977531007329974792387630995534913
196  189 :45039004894146163721691608104044625869634743586863643940090629612726264040939993944
196  190 :89978998798192426443384217108979262738169487283716287980271249225462428190780086998
196  191 :179947007507374852895678434317957524476447983467442585960442497559924857380570074996
196  192 :879417082591133282851472678387542769240837728141868345673877374158183331086270824967
196  193 :1648845155271266664702946456764086637382665466184835591457753648316465662281551539945
196  194 :7148196707093932310841410034306042022199311129022202396134300140391132284007067028406
196  195 :13196404314098754622771820068622974044408522268934404802168600281871264677914143946823
196  196 :46061338456076400840590020754743814488394744849378452724854603099593910466955484415954
196  197 :92012786912042802780189051400586539975789589598766894559600305109098710934020967732018
196  198 :173036563814086704569279201701282089842579188197524888128100720207207431758042936453047
196  199 :913391203054943839271981228703103978268371070172773868410207823180172839438461302083418
196  200 :1727771406219778777543062557405118846645642140346636747711515645369345777787911604276737
```

Marcus

Edited: 9 Mar 2006, 8:24 a.m.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #13 Posted by ***Eamonn*** *on 10 Mar 2006, 12:05 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

Thanks you for another S&SMC. I always find them fun to try, but lately it's been an even tougher challenge to find time to work on them.

I wrote a program in HP-71B Basic to find a solution for the first part of the challenge. I was going to write it in Sharp Basic, but the Sharp handhelds I have only support 10 significant digits and the program needs to be able to handle integers with 11 digits.

I don't have an actual HP-71B, so I used emu71. This was the first time I used this emulator and I agree that it is a great piece of software.

Here is my program. There are some small optimizations to reduce the number of cases that are tested (including starting the outer loop at 150 and working backward - thanks for the hint). However, it still took about 10 minutes to find the answer, even when running with emu71.

```
10 FOR E=150 TO 5 STEP -1 @ E5=E^5 @ FOR D=E-1 TO 4 STEP -1 @ D5=D^5 @ C1=INT ((E5-D5)^.2)
15 IF C1>D THEN D=0 @ GOTO 70
20 FOR C=C1 TO 3 STEP -1 @ C5=C^5 @ B1=INT((E5-D5-C5)^.2) @ IF B1>C THEN C=0 @ GOTO 60
30 FOR B=B1 TO 2 STEP -1 @ B5=B^5 @ A=INT((E5-D5-C5-B5)^.2) @ IF A>B THEN B=0 @ GOTO 50
40 IF A^5+B5+C5+D5=E5 THEN 100
50 NEXT B
60 NEXT C
70 NEXT D
80 NEXT E
90 STOP
100 PRINT A;" ";B;" ";C;" ";D
110 PRINT E
```

It finds the solution 27^5 + 84^5 + 110^5 + 133^5 = 144^5.

I'll see if I can make time to try the other parts of the challenge this weekend.

Best Regards,

Eamonn

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #14 Posted by ***Gerson W. Barbosa*** *on 12 Mar 2006, 1:28 p.m.,*
in response to message #1 by Valentin Albillo

Hello Valentin,

Here is my attempt to cook conjecture #1. Ok, it's in QBASIC, but the conversion to HP-71B basic should be straightfoward. So, not to be completely off the rules, I tested it on the HP-200LX. After 26 minutes and 15 seconds the answer was found:

144 133 110 84 27

The program could be ported to the HP-42S but it would take even longer to run. Perhaps the 49G+ gets the same result in less than 10 minutes. Though the algorithm finds the right answer, it may still have flaws, which everybody is welcome to point me out. Of course, it is easier when we know the answer. If I hadn't seen Eamonn's and other people's solutions, I might not have even tried. Thanks all of you!

As always, I am looking forward to your solutions.

```
5  CLS
10 DEFDBL A-H, O-Z
15 DEFINT I-N
25 DIM Q(150)
30 FOR I = 2 TO 150
35   T = I * I
40   Q(I) = T * T * I
45 NEXT
50 FOR I = 150 TO 6 STEP -1
55    E5 = Q(I)
60    LOCATE 1, 1: PRINT I
65    FOR J = I - 1 TO 5 STEP -1
80      FOR K = J - 1 TO 4 STEP -1
85        FOR L = K - 1 TO 3 STEP -1
87          IF Q(J) + Q(K) > E5 THEN 110
90          FOR M = L - 1 TO 2 STEP -1
91            IF Q(J) + Q(K) + Q(L) > E5 THEN 105
92            S = Q(J) + Q(K) + Q(L) + Q(M)
94            IF S < E5 THEN 110
98            IF S = E5 THEN 130
100         NEXT M
105       NEXT L
110     NEXT K
115   NEXT J
120 NEXT I
130 CLS: PRINT I, J; K; L; M
```

Edited to include Emu71 version:

```
25 STD @ DIM Q(150)
30 FOR I=2 TO 150 @ Q(I)=I^5 @ NEXT I
50 FOR I=150 TO 6 STEP -1
55 E5=Q(I) @ DISP I
65 FOR J=I-1 TO 5 STEP -1
80 FOR K=J-1 TO 4 STEP -1
85 FOR L=K-1 TO 3 STEP -1
87 IF Q(J)+Q(K)>E5 THEN 110
90 FOR M=L-1 TO 2 STEP -1
91 IF Q(J)+Q(K)+Q(L)>E5 THEN 105
92 S=Q(J)+Q(K)+Q(L)+Q(M)
94 IF S<E5 THEN 110 ELSE IF S=E5 THEN 130
```

```
100 NEXT M
105 NEXT L
110 NEXT K
115 NEXT J
120 NEXT I
130 PRINT I,J;K;L;M
```

Since this takes roughly one hour to find the answer (Emu71 @ 500MHz), I think my previous estimation of running time on the 49G+ is wrong: I'd say *less than 10 hours* instead of *less than 10 minutes* :-(

Edited: 12 Mar 2006, 4:20 p.m.

A question about conjecture #1

Message #15 Posted by ***Gerson W. Barbosa*** *on 12 Mar 2006, 5:36 p.m.,*
in response to message #14 by Gerson W. Barbosa

In order to test the algorithm, I made a slight modification in the QBasic program. So beginning with 580, the following results were obtained in about 5 minutes (Pentium III @ 500 MHz, QBX - Compiled QBASIC):

```
576           532  440  336  108
432           399  330  252   81
288           266  220  168   54
144           133  110   84   27
```

Exactly as expected, since (n*a)^5+(n*b)^5+(n*c)^5+(n*d)^5=(n*e)^5. Now, a question arises: besides {133, 110, 84, 24} and their multiples, are there any other four-number sets (and their multiples) that 'cook' the conjecture? Or is this set unique?

I've just found the answer to this question. According to this paper, NEW RESULTS IN EQUAL SUMS OF LIKE POWERS, the set is unique. Perhaps I should attend to an introductory course on Number Theory :-)

Edited: 12 Mar 2006, 6:22 p.m.

Re: A question about conjecture #1

Message #16 Posted by ***Gerson W. Barbosa*** *on 15 Mar 2006, 8:57 p.m.,*
in response to message #15 by Gerson W. Barbosa

I think I had better stick to my first idea:

$\mathbf{85359^5 = 85282^5 + 28969^5 + 3183^5 + 55^5}$ was found by J. Frye (J.-C. Meyrignac, pers. comm., Sep. 9, 2004):

http://mathworld.wolfram.com/DiophantineEquation5thPowers.html

What will come next?

Gerson.

Re: A question about conjecture #1

Message #17 Posted by ***Marcus von Cube, Germany*** *on 16 Mar 2006, 2:16 a.m.,*
in response to message #16 by Gerson W. Barbosa

> Quote:
>
> $\mathbf{85359^5 = 85282^5 + 28969^5 + 3183^5 + 55^5}$

This is even beyond the accuracy of my double precision Sharp PC-E500. The sum has 25 digits while the Sharp can handle 20 digit precision. You need at least 83 bits (unsigned binary) to store a number of that size.

Marcus

Re: A question about conjecture #1

Message #18 Posted by ***James M. Prange (Michigan)*** *on 16 Mar 2006, 3:24 a.m.,*
in response to message #17 by Marcus von Cube, Germany

But verified on the 49 series, in "exact" mode.

Regards,
James

Re: A question about conjecture #1

*Message #19 Posted by **Marcus von Cube, Germany** on 16 Mar 2006, 7:28 a.m.,*
in response to message #18 by James M. Prange (Michigan)

James,

I simply forgot about the CAS machines which come with arbitrary precision integer arithmetic. The TI Voyage 200 can calculate the sum as well.

Marcus

Re: SSMC #14/2 in RPL

*Message #20 Posted by **Marcus von Cube, Germany** on 12 Mar 2006, 5:09 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

here is my attempt at the second problem in RPL on an HP49g+.

First Step: Computing Prime Numbers

The following program, named MKPR, computes a list of primes. The upper limit is given on the stack:

```
%%HP: T(3)A(R)F(.);
\<< { 2. } 'PRIME' STO 3. SWAP
  FOR i 1. \-> j
    \<<
      DO i PRIME j GET
        IF DUP2 SQ \>=
        THEN
          IF / FP
          THEN j 1. + 'j' STO 0.
          ELSE 1.
          END
        ELSE DROP2 PRIME i + 'PRIME' STO 1.
        END
      UNTIL
      END
    \>> 2.
```

```
  STEP
\>>
```

It computes the following list if 2000 is entered on the stack:

{ 2. 3. 5. 7. 11. 13. 17. 19. 23. 29. 31. 37. 41. 43. 47. 53. 59. 61. 67. 71. 73. 79. 83. 89. 97. 101. 103. 107. 109. 113. 127. 131. 137. 139. 149. 151. 157. 163. 167. 173. 179. 181. 191. 193. 197. 199. 211. 223. 227. 229. 233. 239. 241. 251. 257. 263. 269. 271. 277. 281. 283. 293. 307. 311. 313. 317. 331. 337. 347. 349. 353. 359. 367. 373. 379. 383. 389. 397. 401. 409. 419. 421. 431. 433. 439. 443. 449. 457. 461. 463. 467. 479. 487. 491. 499. 503. 509. 521. 523. 541. 547. 557. 563. 569. 571. 577. 587. 593. 599. 601. 607. 613. 617. 619. 631. 641. 643. 647. 653. 659. 661. 673. 677. 683. 691. 701. 709. 719. 727. 733. 739. 743. 751. 757. 761. 769. 773. 787. 797. 809. 811. 821. 823. 827. 829. 839. 853. 857. 859. 863. 877. 881. 883. 887. 907. 911. 919. 929. 937. 941. 947. 953. 967. 971. 977. 983. 991. 997. 1009. 1013. 1019. 1021. 1031. 1033. 1039. 1049. 1051. 1061. 1063. 1069. 1087. 1091. 1093. 1097. 1103. 1109. 1117. 1123. 1129. 1151. 1153. 1163. 1171. 1181. 1187. 1193. 1201. 1213. 1217. 1223. 1229. 1231. 1237. 1249. 1259. 1277. 1279. 1283. 1289. 1291. 1297. 1301. 1303. 1307. 1319. 1321. 1327. 1361. 1367. 1373. 1381. 1399. 1409. 1423. 1427. 1429. 1433. 1439. 1447. 1451. 1453. 1459. 1471. 1481. 1483. 1487. 1489. 1493. 1499. 1511. 1523. 1531. 1543. 1549. 1553. 1559. 1567. 1571. 1579. 1583. 1597. 1601. 1607. 1609. 1613. 1619. 1621. 1627. 1637. 1657. 1663. 1667. 1669. 1693. 1697. 1699. 1709. 1721. 1723. 1733. 1741. 1747. 1753. 1759. 1777. 1783. 1787. 1789. 1801. 1811. 1823. 1831. 1847. 1861. 1867. 1871. 1873. 1877. 1879. 1889. 1901. 1907. 1913. 1931. 1933. 1949. 1951. 1973. 1979. 1987. 1993. 1997. 1999. }

You should set numeric and approximate mode before creating the list. The list is hopefully correct; if not, please complain!

Second Step: Computing Powers

The next RPL program, named MKPOW, creates a list of powers as indicated in the challenge. The upper limit is entered on the stack:

```
%%HP: T(3)A(R)F(.);
\<< { 1. } 'POWR' STO \-> n
  \<< 2. n LN 2. LN /
    FOR p 2. n p XROOT
      FOR i POWR i p ^
        IF DUP2 POS
        THEN DROP2
        ELSE + 'POWR' STO
        END
      NEXT
    NEXT
  \>> POWR SORT 'POWR' STO
\>>
```

Here is the result:

{ 1. 4. 8. 9. 16. 25. 27. 32. 36. 49. 64. 81. 100. 121. 125. 128. 144. 169. 196. 216. 225. 243. 256. 289. 324. 343. 361. 400. 441. 484. 512. 529. 576. 625. 676. 729. 784. 841. 900. 961. 1000. 1024. 1089. 1156. 1225. 1296. 1331. 1369. 1444. 1521. 1600. 1681. 1728. 1764. 1849. 1936. }

If anything is wrong with my list, post a comment, please!

Final Step: Search for Sums

Here is my attempt, named SSMC14, to find a number which cannot be represented as a sum of elements taken from each of the lists:

```
%%HP: T(3)A(R)F(.);
\<< 6. SWAP
  FOR n
    n 3 DISP
    0. \-> p
    \<<
      DO p 1. + 'p' STO n POWR p GET -
        IF DUP 0. >
        THEN PRIME SWAP POS
        ELSE n HALT 1.
        END
      UNTIL
      END
    \>>
  NEXT
\>>
```

If it finds a solution, it hits a breakpoint. The prgramm must be killed manually or can be continued to search for more solutions. The upper limit is entered on the stack. Both lists must be created beforehand.

Here is my result:

1549

Marcus

Re: SSMC #14/2 in RPL

Message #21 Posted by ***Valentin Albillo*** *on 12 Mar 2006, 6:49 p.m.,*
in response to message #20 by Marcus von Cube, Germany

Hi, Marcus:

Your result is correct. Timing ?

Best regards from V.

Re: SSMC #14/2 in RPL

Message #22 Posted by ***Marcus von Cube, Germany*** *on 13 Mar 2006, 2:52 a.m.,*
in response to message #21 by Valentin Albillo

> Quote:
>
> ---
>
> Your result is correct. Timing ?
>
> ---

Wow!

Creating the lists is a matter minutes (PRIME, 2:10) or Seconds (POWR, 0:04). The search itself took 25 minutes. All measurements were done on a 49g+

Marcus

Re: SSMC #14/2 in RPL

Message #23 Posted by ***Arnaud Amiel*** *on 13 Mar 2006, 3:53 a.m.,*
in response to message #20 by Marcus von Cube, Germany

Unfortunately I haven't had time to look at the SSMC but I would have used the NEXTPRIME command on the 49.

It works correctly up to a few millions.

Arnaud

Re: SSMC #14/2 in RPL

Message #24 Posted by ***Marcus von Cube, Germany*** *on 13 Mar 2006, 4:42 a.m.,*
in response to message #23 by Arnaud Amiel

Arnaud,

NEXTPRIME might be worth a try to speed up building the PRIME list but we cannot gain more than about 2 minutes here. On the other hand, the programs posted should work on the lower end machines, too.

The 48S cannot sort the power list but the roles of the two lists in program SSMC14 can be reversed and the prime list is always sorted. This results in a performance penalty because the userRPL code must operate on the longer of the two lists and the built-in operation POS works on the shorter. The 48G has SORT and can use a sorted POWR list.

I'll probably do some tests on my 48 series machines, too.

Marcus

Re: SSMC #14/2 in RPL - Revised

Message #25 Posted by ***Marcus von Cube, Germany*** *on 13 Mar 2006, 5:53 a.m.,*
in response to message #20 by Marcus von Cube, Germany

I revised the above solution slightly and added a controlling program to get some timing information.

MKPOW

```
%%HP: T(3)A(R)F(.);
\<< { 1. } 'POWR' STO
  \-> n
  \<<
    2
    n LN 2 LN /           @ log2 n as upper limit
    FOR p
      2
      n p XROOT           @ n^(1/p) as upper limit
      FOR i
        POWR i p ^
        IF DUP2 POS       @ check if already in list
        THEN DROP2
        ELSE + 'POWR' STO @ add to list
        END
      NEXT
    NEXT
  \>>
  POWR SORT 'POWR' STO    @ sort list (only on 48G and up)
\>>
```

MKPR

```
%%HP: T(3)A(R)F(.);
\<<
  { 2. 3. } 'PRIME' STO   @ set up list with first elements
  5 SWAP                  @ loop from 5 to limit on stack
  FOR i
    2 \-> j               @ j indexes into the list for divisors
    \<<
      DO
        i                 @ number to test
        PRIME j GET       @ get divisor from list
        IF DUP2 SQ \>=    @ check if beyond reasonable limit
        THEN
          IF / FP         @ check if divisible
          THEN
            'j' INCR DROP @ not divisible, try next divisor
            0             @ continue with DO-UNTIL
          ELSE
            1             @ leave DO-UNTIL loop
          END
        ELSE
          DROP2           @ drop i and divisor
          PRIME i +
          'PRIME' STO     @ add new prime to list
          1               @ leave DO-UNTIL loop
        END
      UNTIL               @ flag is already on the stack
      END
    \>>
  2 STEP                  @ next odd number to try
\>>
```

SSMC14

```
%%HP: T(3)A(R)F(.);
\<<
  FOR n                   @ both limits must be on stack!
  n 3 DISP                @ inform user
  0 \-> p                 @ p indexes into the POWR list
    \<<
      DO
        n                 @ number to test
        POWR 'p' INCR GET @ first summand from POWR list
        -                 @ other summand
        IF DUP 0 >        @ test if both summands are positive
        THEN
```

```
          PRIME SWAP POS  @ leave DO-UNTIL if summand is a prime
        ELSE
          DROP n          @ Heureka! Show just the result on the stack
          1E99 'n' STO    @ leave FOR-NEXT
          1               @ leave DO-UNTIL
        END
      UNTIL               @ flag is already on the stack
      END
    \>>
  NEXT
\>>
```

SSMC14-V2
Slower but works with unsorted POWR list on HP48S

```
%%HP: T(3)A(R)F(.);
\<<
  FOR n
  n 3 DISP
  0 \-> p
    \<<
      DO
        n
        POWR 'p' INCR GET
        -
        IF DUP 0 >
        THEN POWR SWAP POS
        ELSE DROP n 1E99 'n' STO 1
        END
      UNTIL
      END
    \>>
  NEXT
\>>
```

TIM
Timing loop

```
%%HP: T(3)A(R)F(.);
\<<
  TIME
  2000 MKPOW TIME 600 .1 BEEP
  2000 MKPR  TIME 600 .1 BEEP
  6 2000 SSMC14 TIME 600 .1 BEEP
\>>
```

The software is currently executing on an HP48G, I'll post timing information later.

Marcus

Here are the execution times on a 48G:

MKPOW: 00:00:16 (h:m:s)
MKPR: 00:09:09
SSMC14: 02:05:10

This is roughly factor 4 compared to the 49g+.

Marcus

Edited: 13 Mar 2006, 8:16 a.m.

Re: Short & Sweet Math Challenge #14: Cooking Conjectures !

Message #26 Posted by ***Gerson W. Barbosa*** *on 12 Mar 2006, 10:34 p.m.,*
in response to message #1 by Valentin Albillo

Here is my Emu71 version for problem #1 slightly modified to avoid some unnecessary additions. Now it is about 50% faster:

```
>LIST
25 STD @ DIM Q(150) @ FOR I=2 TO 150 @ Q(I)=I^5 @ NEXT I
50 FOR I=150 TO 6 STEP -1 @ E5=Q(I) @ DISP I
65 FOR J=I-1 TO 5 STEP -1 @ FOR K=J-1 TO 4 STEP -1 @ FOR L=K-1 TO 3 STEP -1
70 S1=Q(J)+Q(K) @ IF S1>E5 THEN 110
80 FOR M=L-1 TO 2 STEP -1 @ S2=S1+Q(L) @ IF S2>E5 THEN 105
90 S3=S2+Q(M) @ IF S3<E5 THEN 110 ELSE IF S3=E5 THEN 130
100 NEXT M
105 NEXT L
110 NEXT K @ NEXT J
120 NEXT I
130 PRINT I,J;K;L;M
>
>RUN
 150
 149
```

```
148
147
146
145
144
144                    133  110  84  27
```

Running time: 47 min 21 sec @ 500MHz (previously 1 hour 10 min 36 sec).

Regards,

Gerson.

S&SMC#14: My Original Solutions and Comments [LONG]

Message #27 Posted by ***Valentin Albillo*** *on 13 Mar 2006, 6:13 a.m.,*
in response to message #1 by Valentin Albillo

Hi all:

Thanks to all of you who were interested in this S&SMC#14, and most specially to the outstanding ones (Marcus von Cube, Eamonn, Gerson W. Barbosa, Mr. Hohmann) who actually cared to take the time to develop and post their very clever solutions, I really hope you enjoyed it and consider the afforded time well spent. These are my original solutions, with comments.

Conjecture 1: Well-done

This is a particular case of a plausible conjecture issued by **Euler** in 1769, which resisted every effort to prove or disprove it for almost *two centuries* until it was finally disproved in 1966 when the first counterexample was found by Lander and Parkin, using a then quite powerful CDC 6600 computer.

Now, we're asked to duplicate that feat, namely finding a non-trivial solution to $\mathbf{A^5+B^5+C^5+D^5=E^5}$ in non-zero integers, using our favorite handhelds. A straightforward, dumb brute-force search would require *excessive* running times (flattening several sets of batteries in the process), but a few refinements here and there will cut down the task by *several orders of magnitude*. However, these refinements need a significant amount of RAM to implement, which means only advanced models like the HP-41CX, HP42S, HP-71B, and HP48/49 can be profitably used.

Here's my original 7-line, 251-byte solution for the HP-71B

```
1 DESTROY ALL @ OPTION BASE 0 @ K=150 @ DIM P(K)
2 FOR I=0 TO K @ P(I)=I*I*I*I*I @ NEXT I @ R=.2 @ L=2^R @ M=3^R @ N=4^R
3 FOR E=K TO 1 STEP -1 @ T=P(E) @ DISP E
4 FOR D=E-1 TO E/N STEP -1 @ F=T-P(D) @ U=F^R
5 FOR C=INT(U) TO U/M STEP -1 @ G=F-P(C) @ V=G^R @ FOR B=INT(V) TO V/L STEP -1
6 IF NOT FP((G-P(B))^R) THEN A=(G-P(B))^R @ DISP A;B;C;D;E @ END
7 NEXT B @ NEXT C @ NEXT D @ NEXT E
```

upon running, it eventually outputs:

```
>RUN


  27  84  110  133  144
```

which is the sought-for *counterexample*, as indeed

$\mathbf{27^5 + 84^5 + 110^5 + 133^5 = 61917364224 = 144^5}$

The running time is a short 90 min. in an actual, physical HP-71B, and only 13 seconds under Emu71 in my IBM laptop. To achieve such short times the following techniques are used to speed the search, they key of which is to iteratively reduce the problem to a similar yet easier one:

- first of all, a table of 5^{th}-powers for integer arguments from 0 to 150 is *precomputed*, so that during the search the extremely frequent need of raising values to the 5^{th} power is reduced to indexing an array element, which is significantly *faster*.

- some other needed constants are precomputed as well, which saves additional time when they're used inside the search's nested loops, namely 1/5, $2^{1/5}$, $3^{1/5}$, $4^{1/5}$

- the outer loop traverses all possible values for E in *reverse order*, from the largest possible value to the smallest.

- for earch value of E, we must try to decompose E^5 as the sum of four 5^{th}-powers. Assuming D is the *largest* one, we must search for a decomposition of E^5-D^5 in *three* 5^{th}-powers where, as D is the largest element by definition, we only need to search for values of D in the limited range:

  $(E^5/4)^{1/5} <= D < (E^5)^{1/5}$ i.e. $E/4^{1/5} <= D < E$

  which we traverse in reverse order as well. For each value of D, we must now try to decompose E^5-D^5 as the sum of *three* 5^{th}-powers, so *we've reduced our problem to a similar but simpler one*, and the same argument is repeated till a single summand is left, which we simply check to ascertain whether it is a 5^{th}-power or not (i.e.: the fractional part of its fifth root equals 0). When this condition is met, we've found a counterexample so the program outputs it and stops.

Conjecture 2: Medium

We're asked for the smallest integer number from 6 to 2000 that can't be decomposed as the sum of a prime and a non-zero power.

A brute-force search would again require excessive running time, but *we can trade RAM for speed* once more. This is my original 9-line, 259-byte solution for the HP-71:

```
1 DESTROY ALL @ OPTION BASE 1 @ M=2000 @ D=LOG(M) @ N=INT(SQR(M))
2 INTEGER P(M),Q(310) @ FOR B=2 TO N
3 FOR E=2 TO INT(D/LOG(B)) @ P(B^E)=1 @ NEXT E @ NEXT B @ P(1)=1 @ Q(1)=2 @ I=2
4 FOR N=3 TO M-1 STEP 2 @ FOR D=3 TO INT(SQR(N)) STEP 2 @ IF MOD(N,D)=0 THEN 6
5 NEXT D @ Q(I)=N @ I=I+1
6 NEXT N @ T=I-1 @ FOR N=6 TO M
7 FOR D=1 TO T @ E=N-Q(D) @ IF E<=0 THEN DISP N @ END ELSE IF P(E) THEN 9
8 NEXT D @ DISP N @ END
9 NEXT N
```

upon running, it outputs:

```
>RUN


    1549
```

which is the *smallest counterexample*. The running time is a fast 30 min. in a physical HP-71B, and only 6 seconds under Emu71 in my IBM laptop, thanks to these simple yet efficient techniques:

- first of all, this challenge's actually far easier than it seems at first. Actually, for the search to proceed fast and smoothly, we need but two things: (1) to have all relevant prime numbers instantly at hand, and (2) to be able to test *extremely quickly* whether a given integer is a perfect power or not. This is achieved as follows:

- all primes up to 2000 are *precomputed and stored* in an array so that they can be retrieved very fast during the search

- in order to quickly check whether a given integer is a perfect power or not, we stablish a 2000-element *'flag'* array where each element has the value 1 or 0 depending on whether the index is a perfect power or not, i.e., P(7)=0, P(8)=1 (as $8=2^3$), etc. This could be achieved using much less RAM by packing eight such 'flags' per byte in a string, for instance, but here speed is our top priority so we simply define an integer array and let each individual element act as a 'flag'.

- with primes and very fast power-detection available, the main search just traverses all possible values from 6 to 2000, in ascending order. For each, every prime is subtracted in turn and the result is checked to see if it is a perfect power, skipping to the next value if it is. When no prime subtracted will result in a perfect power remaining, the corresponding value is output as a valid *counterexample*.

- by the way, the prime array is dimensioned to have 310 elements because there are some 300+ prime numbers in the range from 2 to 2000 (actually, 303). If you've got INTEGRATE capabilities in your HP model, you can very quickly compute an *approximate* value for the required number of elements as follows:

```
approx. #primes up to M = INT(INTEGRAL(2,M,1,1/LN(IVAR))
```

so that the line:

```
2 INTEGER P(M),Q(310) @ ...
```

becomes:

```
2 INTEGER P(M),Q(INTEGRAL(2,M,1,1/LN(IVAR))) @ ...
```

and this allows you to expand the search to numbers greater than 2000 by simply changing the value assigned to M at line 1, without having to consult tables for the proper size of array Q. As the integration's *uncertainty* is specified as 1, (i.e: FIX 0 or SCI 0 for models that use the display setting instead), the integral is computed *very quickly*.

Now, you'll agree with me that a line which dimensions an array with a size *defined by a non-elementary integral* isn't that frequent a sight.

Conjecture 3: Rare

This is a well-known conjecture *as yet unproved or disproved*, though everybody and his uncle feels that **196** is a true counterexample, as it has failed to produce a palindrome after hundreds and hundreds of millions of cycles. For instance, after 670,000,000 cycles you're dealing with numbers *280,000,000-digit long*(!!), yet no palindrome in sight ...

In order to implement this challenge, *multiprecision addition* is mandatory, which can be done with arrays or else with strings, which is the technique I've used in my original 8-line, 364-byte solution for the HP-71B:

```
1 DIM M$[500],N$[500] @ INPUT "#Cmax=";M @ FOR I=0 TO 200 @ N$=STR$(I) @ K=0
2 CALL MADD(N$,M$,K) @ IF M$=REV$(M$) THEN 4 ELSE IF K<M THEN N$=M$ @ GOTO 2
3 DISP I;"fails (";M$;" after";M;"cycles)"
4 NEXT I @ DISP "OK" @ END


5 SUB MADD(A$,C$,K) @ DIM B$[500] @ L=LEN(A$) @ E=10^11 @ C$="" @ C=0
6 B$=REV$(A$) @ FOR I=L TO 1 STEP -11 @ C=VAL(A$[I-10,I])+VAL(B$[I-10,I])+C
7 D$=STR$(MOD(C,E)) @ C$=RPT$("0",11-LEN(D$))&D$&C$ @ C=C DIV E @ NEXT I
8 C$=LTRIM$(STR$(C)&C$,"0") @ K=K+1 @ END SUB
```

The algorithm is completely straightforward and no special techniques are needed, though both for speed and to avoid obscuring the inner works with trivial utility routines, I've made use of several string-handling keywords (REV$, RPT$, LTRIM$) which are available in a number of very common LEX files and ROMs (STRNGLEX, REVLEX, RPTLEX, JPC ROM, etc). The program works as follows:

- the multiprecision results will be stored in strings (which in the case of the HP-71B would allow us to handle *up to 65000-digit numbers!*), initially dimensioned to hold up to 500-digit numbers, more than enough for up to 1000 cycles.
- to perform the multiprecision addition of a number and its mirror image, a call to the MADD subprogram is made. This subprogram takes the number as one of its string arguments pased by value, and returns the result of the addition as another string argument passed by reference. For instance:

  ```
  >CALL MADD("78",M$,0) @ M$

    165

  >>CALL MADD("8263485213753244473212",M$,0) @ M$

    10387229637326370316840
  ```

  because 78+87 = 165 and 8263485213753244473212+2123744423573125843628 = 10387229637326370316840. Actually, the subprogram's code could be inserted directly in the main program proper, so we could get rid of the subprogram altogether and get a slightly faster, shorter program (7 lines instead of 8) but 'outsourcing' particular tasks to subprograms encourages modular programming and makes for clearer, cleaner code and provides additional functionality as well.

  After calling the subprogram, the main program just checks if the new result is *palindromic*, or if we've exhausted the number of cycles, and iterates as needed.

Upon running, it produces the following, for assorted maximum number of cycles (10, 20, 40, 100, 200, 1000):

```
>RUN

#Cmax=10

  89 fails (8872688 after 10 cycles)
  98 fails (8872688 after 10 cycles)
 167 fails (17050517 after 10 cycles)
 177 fails (17794887 after 10 cycles)
```

```
 187 fails (17735476 after 10 cycles)
 196 fails (18211171 after 10 cycles)

OK

>RUN

#Cmax=20

  89 fails (93445163438 after 20 cycles)
  98 fails (93445163438 after 20 cycles)
 187 fails (176881317877 after 20 cycles)
 196 fails (70446464506 after 20 cycles)

OK

>RUN

#Cmax=40

 196 fails (13305261530450734933 after 40 cycles)

OK

>RUN

#Cmax=100

 196 fails (44757771534490515617290699271561508443627774644 after 100 cycles)

OK

>RUN
```

```
#Cmax=200

 196 fails (910449546741765655298269802255629632301207255281
           2103235826563197972803556567037646054008 after 200 cycles)

OK

>RUN

#Cmax=1000

  196 fails (35346644392413689785837714402912114362859098083
             41408344020861450405992918328457190349563871687
             95800463971545914548326676428378028814710683108
             50549641273388365259932008237493462055424091251
             57901200166876923521977766210101074152201325440
             26439582289914006246477437313605494900387117318
             73088382467552483640965506947400858697069355944
             18174493380829951504425811945379423290791058264
             41012030342772858788740429334664452 after 1000 cycles)

OK
```

So it's clear that **196** is a very firm *candidate* for a counterexample. Of course, it's not the only one, other candidates include (up to N=2000): 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986, 1495, 1497, 1585, 1587, 1675, 1677, 1765, 1767, 1855, 1857, 1945, 1947, and 1997.

That's all. Thanks again and

Best regards from V.

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #28 Posted by ***J-F Garnier*** *on 14 Mar 2006, 3:33 a.m.,*
in response to message #27 by Valentin Albillo

Hello Valentin,

Unfortunatly, I had little time for your challenge, but the first part looked especially interesting for me, trying to reproduce a proof made on 1966 on a large computer with a small handheld of the '80s.

And your solution is so simple, as usual ...

Do you have a reference to the original paper from Lander and Parkin? I would like to know more on how they proceeded.

J-F

Re: S&SMC#14: My Original Solutions and Comments [LONG]

*Message #29 Posted by **Valentin Albillo** on 14 Mar 2006, 5:41 a.m.,*
in response to message #28 by J-F Garnier

Hi, Jean-François:

Jean-François posted:

"Unfortunatly, I had little time for your challenge, but the first part looked especially interesting for me, trying to reproduce a proof made on 1966 on a large computer with a small handheld of the '80s."

Yes, I missed your usual top-quality contributions to my challenges, it's a real pity you couldn't give it a try. And I agree on the fun factor of having a quite petite handheld doing essentially the same as an original mainframe computer which used to fill a whole room:

The awesome CDC 6600 mainframe computer

This CDC 6600 was designed by Seymour Cray himself, of Cray supercomputers fame, and had a 60-bit processor capable of 4.58 Mflops, impressive for its time indeed.

"And your solution is so simple, as usual ..."

Thanks, J-F, but I insist, my challenges *are* simple, that's why I call them "Short & Sweet". They never require more than a few lines of code, and simple code at that.

"Do you have a reference to the original paper from Lander and Parkin? I would like to know more on how they proceeded."

Reference, yes:

"Lander, L. J. and Parkin, T. R. "A Counterexample to Euler's Sum of Powers Conjecture." Math. Comput. 21, 101-103, 1967."

Unfortunately, the paper itself, no. It seems to be available only from non-free subscription services. As far as I know, their procedure was similar to mine, only more ambitious because they also attacked more general equations, involving powers up to the 6th. It's quite possible for they to have made use of congruences (i.e. particular remainders when using specific divisors, etc) to discard ranges of tentative values for the inner loops.

I did try a few congruences but the program got more complicated and worse, no speed was actually gained. In a CDC 6600 assembly-language program or compiled code, using just integer operations, congruences would be very effective. But for the floating point HP-71B environment, no gains are made in this case.

Thanks for your interest, and please don't miss my next challenge, come next April 1st.

Bes regards from V.

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #30 Posted by ***Marcus von Cube, Germany*** *on 14 Mar 2006, 9:01 a.m.,*
in response to message #29 by Valentin Albillo

Hi Valentin,

> Quote:
>
> ... and please don't miss my next challenge, come next **April 1st**

Can we expect an April's fool joke?

Marcus

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #31 Posted by ***Etienne Victoria*** *on 14 Mar 2006, 11:14 a.m.,*
in response to message #28 by J-F Garnier

Hello Jean-François,

And don't miss the Euler.net for interesting stuff on conjectures.

Cordialement.

Etienne

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #32 Posted by ***GE*** *on 15 Mar 2006, 7:57 a.m.,*
in response to message #31 by Etienne Victoria

Thank you for an interesting challenge.
I was disappointed to see that no actual math trick could be used here, that was rather a programmer's challenge.
Please accept my apologies for posting a solution without code, I was just thrilled to 'know' the solution of one of the challenges.
Actually I tried Challenge 1 on my 1BBBB, and it ran about the same as solutions showed her, but with no optimizations and I felt it was too slow and ridiculous to be posted here.
To make for my inappropriate post, here is a small challenge for you : solve 28^x = 19^y + 87^z for x, y and z integers.
There IS some math trickery possible...
See you on the 1st of April.

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #33 Posted by ***J-F Garnier*** *on 21 Mar 2006, 6:38 a.m.,*
in response to message #32 by GE

Hi GE,

28^x = 19^y + 87^z for x, y and z integers

You didn't give us the solution...

J-F

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #34 Posted by ***Valentin Albillo*** *on 21 Mar 2006, 6:53 a.m.,*
in response to message #33 by J-F Garnier

Hi, Jean-François:

Jean-François posted:

"Hi GE, 28^x = 19^y + 87^z for x, y and z integers You didn't give us the solution..."

> There is *no* solution to the equation you mention in your post, so it's no big surprise the person you're addressing would fail to provide one.
>
> Per my posted commitment I read or answer nothing by anonymous posters so I don't quite know what's all about. Some kind of joke, perhaps ?

Best regards from V.

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #35 Posted by ***Chris Dean*** *on 21 Mar 2006, 8:11 a.m.,*
in response to message #34 by Valentin Albillo

Valentin et al

I had a go at this myself with no joy

> Quote:
>
> ---
>
> See you on the 1st of April.
>
> ---

Could this have been a clue!

Regards

Chris

Re: S&SMC#14: My Original Solutions and Comments [LONG]

Message #36 Posted by ***Valentin Albillo*** *on 21 Mar 2006, 8:36 a.m.,*
in response to message #35 by Chris Dean

Hi, Chris:

Chris posted:

"I had a go at this myself with no joy. Quote: See you on the 1st of April. Could this have been a clue!"

> Can't say. As stated, I didn't read the original post J-F mentions, so I don't know what was asked or any further comments about it, though judging from J-F's post, it seems that a solution was asked to a diophantine equation which has none.
>
> It that was indeed the case, how this is supposed to be considered enjoyable by people wasting their time and computer resources on an impossible task is beyond me. Perhaps the mere idea of having people indulge in fruitless efforts seemed indeed enjoyable to the original poster, but I can't say for sure, who knows ...

Best regards from V.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall