HP Forum Archive 15

[Return to Index | Top of Index]

Short & Sweet Math Challenges #8: Squares !

*Message #1 Posted by **Valentin Albillo** on 20 Apr 2005, 10:09 a.m.*

Hi all, long time no see:

After many months have elapsed since I posted the last S&SMC, here's a new one which certainly lives up to its title. Should you feel like it, grab your favorite HP calculator(s) (any programmable one will do), then try and solve this

Challenge:

1. Find a **10-digit square** which consists of just **two different** numbers, from 1 to 9 (0's not allowed). For example, this would be a solution (featuring just the digits 1 and 3):

   ```
   3113311313
   ```

   if said number were a square which, alas, it isn't.

   The solution is unique. You're expected to produce a **program** for your chosen calculator(s) that upon running will find a solution, also making sure there are no others. If you succeed, you may want to extend your program to also:

2. Find **all** 10-digit squares which consist of at most **three** different numbers, from 1 to 9 (0's not allowed). For example, this would be a solution, (featuring just the digits 3,5, and 7):

   ```
   5733753373
   ```

   if it were a square, which of course it isn't. There are 48 solutions in all.

As stated, you must produce a program for your HP calculator(s) that will compute and output the solution(s), the shorter and faster the better. Giving just the solutions or providing programs for machines other than HP calculators is very nice of you, thank you, but you simply forfeited the challenge, you loser ! ;-)

Next Friday I'll post the solutions and a 4-line program for a bare bones HP-71B which promptly finds out the solutions. Meanwhile be a sport and try your best. Remember, it's one thing to boast about your efficiency with RPN or RPL or whatever, and quite another to deliver the goods when challenged ... ;-)

Best regards from V.

Edited: 22 Apr 2005, 7:20 a.m. after one or more responses were posted

My brain hurts

*Message #2 Posted by **Gene** on 20 Apr 2005, 10:39 a.m.,*
in response to message #1 by Valentin Albillo

:-)

C'mon, Gene ! It's quite simple :-)

*Message #3 Posted by **Valentin Albillo** on 20 Apr 2005, 10:48 a.m.,*
in response to message #2 by Gene

In your case I'll gladly make an exception: do post a program for a TI machine, should you feel more comfortable with it.

I know you're a keen defender of TI machines, just as I am of (some) SHARP ones, so this might be your opportunity to show that they're up to the task.

Best regards from V.

Re: C'mon, Gene ! It's quite simple :-)

*Message #4 Posted by **Gene** on 20 Apr 2005, 10:51 a.m.,*
in response to message #3 by Valentin Albillo

Insults? :-)

I like the old TI machines, but I am an RPN / RPL man!

I can't imagine trying to write a program to do this on a 100-step SR-56. :-)

Of course, that's primarily because I haven't thought through the problem enough yet, I suppose!

Gene

Re: C'mon, Gene ! It's quite simple :-)

*Message #5 Posted by **Valentin Albillo** on 20 Apr 2005, 12:24 p.m.,*
in response to message #4 by Gene

Hi again, Gene:

Gene posted: *"Insults? :-)*

No, why on earth would you feel insulted ? Myself, I've stated a number of times that vintage SHARP machines are every bit as good if not better than contemporary HP ones and I wouldn't feel insulted in the least if someone were to offer me using a SHARP instead of an HP. But a TI ... that's a horse (donkey ?) of a different color ... come to think of it, you have a point, my apologies.

"I like the old TI machines, but I am an RPN / RPL man!"

I'm *not*.

"I can't imagine trying to write a program to do this on a 100-step SR-56. :-)"

I don't know about the SR-56, but I've got a quick'n'dirty 41CX version in some 60 steps ... Way to go.

Now stop ranting and do write a solution, man ! :-)

Best regards from V.

SR-56 memories...

*Message #6 Posted by **Marcus von Cube** on 23 Apr 2005, 11:17 a.m.,*
in response to message #4 by Gene

> Quote:
>
> I can't imagine trying to write a program to do this on a 100-step SR-56.

My flag (re)setting algorithm will certainly not work! The most complicated problem I've ever tackled on the 56 was the solving of 3 linear equations with 3 unknowns. Because there are only 10 registers on the machine, I had to enter the constants on the right hand side of the equations 3 times (once for each unknown.)

And of course, I had to type in the whole program every time I wanted to use it...

... and a link to the HP-20S

*Message #7 Posted by **Karl Schneider** on 23 Apr 2005, 1:37 p.m.,*
in response to message #6 by Marcus von Cube

Marcus von Cube posted,

> Quote:
>
> The most complicated problem I've ever tackled on the [Texas Instruments TI-]56 was the solving of 3 linear equations with 3 unknowns. Because there are only 10 registers on the machine, I had to enter the constants on the right hand side of the equations 3 times (once for each unknown.)

The HP-20S (discontinued in 2002), also has only 10 storage registers. It includes a built-in loadable keystroke program for solving an exactly-determined 3x3 set of linear equations, using Cramer's Rule.

After the program "3by3" is loaded into program memory, the user enters the 9 matrix coefficients in registers 1-9, and the "upper" result constant into register 0. The remaining two constants are placed in the stack, separated by "INPUT". "XEQ A" calculates the solution.

The program can be used to solve an exactly-determined 2x2 system by entering the system as follows:

```
[a11 a12 0]  [b1]
[a21 a22 0]  [b2]
[ 0   0  1]  [ 0]
```

A matrix can be inverted by solving for $[1\ 0\ 0]^T$, $[0\ 1\ 0]^T$, and $[0\ 0\ 1]^T$ in succession.

Determinants can be calculated using "XEQ D" after loading the "3by3" program.

Kind of a pain in many respects, but it's better than nothing, I suppose...

-- KS

Re: Short & Sweet Math Challenges #7: Squares !

*Message #8 Posted by **Larry Corrado, USA (WI)** on 20 Apr 2005, 9:03 p.m.,*
in response to message #1 by Valentin Albillo

Valentin:

Rather than RPN or RPL, I used "whatever." Namely, Java on my Dell. Does that count? ;-)

I love my HP-25, which I used daily for 25 years. My HP-15C is beautiful, with its crisp, hi-contrast display. I love the glowing digits on my 34C... But when it comes to programming, I'll stick to Java or VB.NET.

I look forward to seeing your 4-line solution. Thanks for presenting the challange.

Ciao, Larry

Re: Short & Sweet Math Challenges #7: Squares !

*Message #9 Posted by **Arnaud Amiel** on 21 Apr 2005, 4:40 a.m.,*
in response to message #1 by Valentin Albillo

This is short but dirty and slow (just over 1 hour) on the 49+. It will not work in RPN though...

```
<< RCLF {} 'RESULTS' STO 10. STWS BIN
1. 8. FOR I #1b I 1. + 9.
  FOR J 1. 1022.
    FOR K DUPDUP NOT ->STR TAIL OBJ-> DROP I * SWAP ->STR TAIL
          OBJ-> DROP J * + DUP
          IF FP THEN DROP ELSE 'RESULTS' STO+ END
    #1b +
    NEXT
  NEXT DROP
NEXT STOF
>>
```

And it tells me that 6661661161 is the only answer.

Now I have to find an other more efficient way to do it.

Arnaud

Erratum

*Message #10 Posted by **Arnaud Amiel** on 21 Apr 2005, 5:53 a.m.,*
in response to message #9 by Arnaud Amiel

I forgot to type a SQRT (well a square root sign) just before the IF

Appart from that it works but so slowly... this is the kind of algorithm that would do well in assembly. Oh and I can't use this algorithm to go to step 2.

Arnaud

Edited: 21 Apr 2005, 6:21 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

*Message #11 Posted by **Jeff O.** on 21 Apr 2005, 6:34 p.m.,*
in response to message #1 by Valentin Albillo

Without looking at Larry's or Arnaud's responses (I swear!) , in case they gave the answers, I'll go out on a limb and present the following:

> Quote:
>
> 1. Find a 10-digit square which consists of just two different numbers, from 1 to 9 (0's not allowed)....The solution is unique.

According to my hp-42S program, that number would be 6661661161, with a square root of 81619.

> Quote:
>
> 2. Find all 10-digit squares which consist of at most three different numbers, from 1 to 9 (0's not allowed).

According to my hp-42S program, the 10 digit squares that consist of **just** three different numbers are as follows:

```
COUNT   NUMBER  SQUARE
1       33334   1111155556
2       33335   1111222225
3       33338   1111422244
4       33359   1112822881
5       33989   1155252121
6       34641   1199998881
7       34827   1212919929
8       34965   1222551225
9       36361   1322122321
10      37962   1441113444
11      39388   1551414544
12      39581   1566655561
13      40204   1616361616
14      43923   1929229929
15      44623   1991212129
16      47162   2224254244
17      50235   2523555225
18      54123   2929299129
19      54335   2952292225
20      57665   3325252225
21      58167   3383399889
22      62156   3863368336
23      66515   4424245225
```

```
24      66665   4444222225
25      66667   4444488889
26      66668   4444622224
27      67485   4554225225
28      68187   4649466969
29      68962   4755757444
30      70107   4914991449
31      70173   4924249929
32      72285   5225121225
33      72335   5232352225
34      72475   5252625625
35      76478   5848884484
36      78196   6114614416
37      78541   6168688681
38      78881   6222212161
39      79162   6266622244
40      80408   6465446464
41      81346   6617171716
42      81404   6626611216
43      81813   6693366969
44      83666   6999999556
45      86478   7478444484
46      97773   9559559529
47      98336   9669968896
```

I was a bit distressed that I found only 47 of these, as apposed to the 48 that V. stated exist. However, upon reading his challenge, it does say *at most*. The group should therefore include those that have *just* 3 numbers listed above, plus the solution that has just 2 numbers, plus any that consist of just one number. A quick check of 111111111, 222222222, etc. with my calculator revealed that none of these are perfect squares. So the list of 47 above plus the unique solution to the first problem yields the 48 solutions to the second problem. (I hope.)
I'm quite sure that I used a very non-elegant, non-optimized, brute-force method. The program I wrote consisted of several basic parts.
1. create the 10 digit numbers by squaring all possible numbers that could create them. The maximum 10 digit number that meets the first criterion is 999999998, whose square root is 99999.99999. So no need to check anything above 99999. The minimum that satisfies the first criterion is 1111111112, whose square root is 33333.33333, so no need to check anything lower than 33334.
2. Break the 10 digit number up into 10 single digits stored in registers 1 through 10 (checking for numbers with zeroes and throwing them out along the way).
3. sort those digits in ascending order in registers 1 through 10.
4. run through registers 1 through 10 and count the number of times that two adjacent digits are not equal to each other.
5. See if the above is either 2, to solve the first problem or 3 to solve the second. If so, print out the number and the square.
6. Move on to the next number.

The program I wrote based on the above methodology was an HP-42S program. However, I developed and ran it on Free42. On my PC, my first version solved each of the problems in about 14 seconds. I then optimized the sorting and checking routines somewhat, and it solved each problem in about 9 seconds. I entered that version into a real 42S. After about 3 hours and 45 minutes I stopped it to make sure it was working correctly. It had found the first 8 solutions, and was about 2.4% of the way through the sequence of numbers. Based on that, it would have taken nearly 6 days to complete. Obviously, I chose a quite inefficient solution scheme. A different approach would be required to make it realistically soluable by a real 42S.

Edited: 25 Apr 2005, 7:16 a.m. after one or more responses were posted

Re: Short & Sweet Math Challenges #7: Squares !

*Message #12 Posted by **Jonathan Puvis (New Zealand)** on 21 Apr 2005, 9:57 p.m.,*
in response to message #11 by Jeff O.

I came up with pretty much the same algorithm (before you posted your message). I initially wrote it in Python (attached below), but my User-RPL version on the 48G ran for more than an hour after which i stopped it as the low battery warning had come one.

I could write the digit counting function in Saturn assembly to speed it up, but i'm sure there is a better algorithm for counting the digits, i just can't think of it.

```
#!/usr/bin/python

def count_digits(x):
  x = int(x)
  digits = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  while x:
    digits[x % 10] = 1
    x = x / 10
  if digits[0]:
    return 99
  return sum(digits)

for x in range(33333, 99999):
  sq = x * x
  if count_digits(sq) == 2:
    print '2:', x, sq

print

for x in range(33333, 99999):
  sq = x * x
  if count_digits(sq) <= 3:
    print '<=3', x, sq
```

HP-32S solution

*Message #13 Posted by **Eamonn** on 21 Apr 2005, 11:32 p.m.,*
in response to message #1 by Valentin Albillo

Valentin,

It's been a while since you posted a S&SMC! As ever, they are quite thought provoking.

I had to have a go at writing a program for the HP-32S to solve the first problem.

The basic idea was to generate all the possible ten-digit numbers that consist of two different numbers, take their square root and test to see if the fractional part is non-zero. There are C(9,8) * 2^10 = 36864 such numbers.

My first attempt would have taken about 10 hours to generate and test all such numbers (It generated and tested about 1 number per second). I won't bother posting it.

After thinking about it a bit, I took a different approach to generating the numbers that generates and tests all of them in about 27 minutes (almost 23 numbers per second).

This second approach uses almost all the 390 bytes of memory on the HP-32S. I'm guessing that it will also run on the HP-33S - not sure if it will be much faster or slower. Perhaps someone would be willing to try it out.

A different approach is needed for the second part of the problem - if my math is correct, there are almost 5 million possibilities for a 10-digit number made up of 3 unique digits.

Anyway, here is the program that I wrote. The program can be run by hitting XEQ V. It stops and displays each solution it finds. The user needs to hit the R/S key to continue after each solution. The number of solutions found is stored in the E register. It finds one solution for the first problem (6661661161 = 81619^2).

Eamonn

```
LBL T   ; Test the value that is in C
RCL C
SQRT
FP
X<>0?
RTN
1
STO+ E ; Count of how many solutions were found
RCL C
STO D  ; Store most recent solution
R/S    ; Stops to display latest solution - User needs to hit R/S to continue
RTN
[CHKSUM = A889]


LBL A
XEQ T
RCL J
STO+ C
XEQ T
RCL K
STO+ C
XEQ T
RCL J
STO+ C
XEQ T
RTN
[CHKSUM = 5BA1]


LBL B
XEQ A
RCL L
STO+ C
XEQ A
RCL M
STO+ C
XEQ A
RCL L
STO+ C
XEQ A
RTN
[CHKSUM = 3AA5]


LBL C
XEQ B
RCL N
STO+ C
XEQ B
RCL O
STO+ C
XEQ B
RCL N
STO+ C
XEQ B
```

```
RTN
[CHKSUM = D82D]


LBL D
XEQ C
RCL P
STO+ C
XEQ C
RCL Q
STO+ C
XEQ C
RCL P
STO+ C
XEQ C
RTN
[CHKSUM = 1A38]


LBL E
XEQ D
RCL R
STO+ C
XEQ D
RCL S
STO+ C
XEQ D
RCL R
STO+ C
XEQ D
RTN
[CHKSUM = EEA7]


LBL V ; <---- Program Entry Point
0
STO E ; Initialize number of answers
9
STO A ; First Digit
[CHKSUM = 8B6D]


LBL F
RCL A
1
-
x=0?  ; If B will be zero, then we're done
RTN   ; <---- Program Exit Point
STO B ; Second Digit
[CHKSUM = 60AB]


LBL G
11.019
STO i
1
STO J
[CHKSUM = 44BD]
```

```
LBL H ; Setting up constants used by the program
10
*
1
-
STO (i) ; K(i) = 10 * K(i-1) - 1
ISG i
GTO H
10.019
STO i
RCL A
RCL- B
[CHKSUM = AF37]


LBL I  ; Loop to multiply constants by (A-B)
STO* (i)
ISG i
GTO I
1111111111
RCL *B
STO C  ; C is the first value to test
XEQ E  ; Call recursive generation of test values
DSE B  ; Loop on second digit
GTO G
DSE A  ; Loop on first digit
GTO F
[CHKSUM = 62F8]
```

Re: HP-32S solution

*Message #14 Posted by **Arnaud Amiel** on 24 Apr 2005, 3:14 a.m.,*
in response to message #13 by Eamonn

> Quote:
>
> This second approach uses almost all the 390 bytes of memory on the HP-32S. I'm guessing that it will also run on the HP-33S - not sure if it will be much faster or slower. Perhaps someone would be willing to try it out.

It takes about 19min on the 33s that is about 33% faster.

Arnaud

Re: Short & Sweet Math Challenges #7: Squares !

*Message #15 Posted by **Bram** on 22 Apr 2005, 7:03 a.m.,*
in response to message #1 by Valentin Albillo

Hi V,

1. Thanks for the challenge

2. Did you loose track? Think there already has been a #7
http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv014.cgi?read=59918

3. My program is for the 32SII. After a short analysis it appeared that "only" the numbers between 31622 and 99999 are eligible for an answer.I check them all for the criteria and I break out the moment a criterion fails. That's the only speed up, so finding the answer 81619 takes several hours.
The program is for the 3 dupl. problem; if you remove the marked instructions, you get the program for the single solution problem.

```
LBL A   ; CK=1BB7 012.5
31622
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C
RCL N
99999
x>=y?
GTO B
STOP


LBL C   ; CK=BA5C 013.5
FS? 0
PSE
CF 1
CF 2
CF 3    ; <=
STO R
10      ; squared number consists of 10 digits
STO I
LBL J   ; CK=C11A 021.0
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed, next number
FS? 1   ; digit already encountered?
GTO K   ; yes
STO X   ; no, store it for comparison with more digits
SF 1    ; digit now has encountered
GTO M
LBL K   ; CK=CDA4 015.0
RCL X   ; get first unique digit
x=y?    ; equal to the one in question?
GTO M
x<>y
FS? 2   ; like before, but now for the second digit
GTO L
```

```
STO Y
SF 2
GTO M
LBL L   ; CK=B38A 015.0
RCL Y
x=y?
GTO M
x<>y    ; <=
FS? 3   ; <=    like before, but now for the third digit
GTO N   ; <=
STO Z   ; <=
SF 3    ; <=
GTO M   ; <=
LBL N   ; CK=2764 007.5
RCL Z   ; <=
x=y?    ; <=
GTO M   ; <=
RTN
LBL M   ; CK=102A 010.5
DSE I
GTO J   ; go to the next digit to examine
RCL N   ; all digits examined, so now we have an answer
x²
STOP
RTN



total length:     119.5


;        storage registers
;        N      number to examine
;        R      number² being split into digits
;        X Y (Z) the digit to occur
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        JKL(N) place holders
;        M      increment to the next digit to examine
;
;        flags
;        0      set => show progression
;        12(3)  keep track of new occurrance of digit
```

Thanks a lot, Bram !

*Message #16 Posted by **Valentin Albillo** on 22 Apr 2005, 7:25 a.m.,*
in response to message #15 by Bram

Hi, Bram:

Bram posted:

"2. Did you loose track? Think there already has been a #7 http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv014.cgi?read=59918 "

Yes, I certainly did. Though I searched my files to see what the current number should be, it seems I didn't keep a record of that particular S&SMC (#7), so I thought that #6 was the very last to date.

I've duly corrected the numbering, so that it won't be kept in the MoHP Archives section with the wrong, duplicated number.

Thanks a lot, see my post with the solutions and comments within 12 hours.

Best regards from V.

Re: Thanks a lot, Bram !

*Message #17 Posted by **Bram** on 22 Apr 2005, 10:28 a.m.,*
in response to message #16 by Valentin Albillo

Anytime Valentin, and in the meantime I rewrote my program to a more versatile one based on indexing. The marked line defines the criterion for the number of digits. Change the 3 into 2 for the first problem. Changing the comparison in the next line is for adoption to exactly x different digits or a least y different digits a.s.o.

regards,
Bram

```
LBL A   ; CK=1BB7 012.5
31622
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C
RCL N
99999
x>=y?
GTO B
STOP


LBL C   ; CK=78F9 009.0
FS? 0
PSE
STO R
9
STO i
LBL I   ; CK=DD00 010.5
0
STO (i)
DSE i
GTO I
10      ; squared number consists of 10 digits
STO J
LBL K   ; CK=0B4E 027.0
```

```
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed, next number
10
*
STO i
1
STO+ (i)
DSE J
GTO K
10
STO i
LBL J   ; CK=E621 022.5
RCL (i)
x#0?
1
STO+ J
DSE i
GTO J
3       ; <= max. number of different digits
RCL J
x>y?
RTN
RCL N
x²
STOP
RTN



total length:     106.0


;        storage registers
;        N      number to examine
;        R      number² being split into digits
;        A-I&J  for counting occurrences
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        IJK    place holders
;
;        flags
```

Re: Third version

*Message #18 Posted by **Bram** on 26 Apr 2005, 4:44 p.m.,*
in response to message #17 by Bram

Hi Valentin, once more....

My second program was more versatile, but much slower than my first one. I tried to speed up my original program by squeezing the inner part to a minimum number of instructions. In only 15 lines (starting at ENTER) I test all criteria and you may be interested in how I use "conditional comparison" to get this done.
I hope things are clear enough by my comments in the program.

```
LBL A   ; CK=1BB7 012.5
31622   ; smallest root of ten digits number minus 1
STO N
LBL B   ; CK=35F8 024.5
1
STO+ N
RCL N
x²
XEQ C   ; examine all ten digits squares
RCL N
99999
x>=y?
GTO B
STOP


LBL C   ; CK=AB00 015.0
FS? 0   ; wish to see progression?
PSE     ; yes, show square that's going to be examined
STO R
0
STO X   ; clear x Y Z on behalf of the three different digits
STO Y
STO Z   ; <=
10      ; squared number consists of ten digits
STO I
LBL J   ; CK=9B2F 037.5
RCL R   ; go strip off a single digit
IP
10
/
STO R
FP
x=0?
RTN     ; no zeros allowed => next number
ENTER   ; put the digit to examine into Y for comparisons
x<>X    ; save it in X and get former value from X
x#0?    ; if former value was 0, we have a new digit; goto M for next digit
x=y?    ; if equal, we have a duplicate digit; goto M for next digit
GTO M
x<>X    ; digit is a new, second one; restore X with original contents
        ; and get back the digit to examine into X
x<>Y    ; same as before; check against 2nd distinct digit that has occurred
x#0?
x=y?
GTO M
x<>Y    ; <= C
x<>Z    ; <=  L
x#0?    ; <=   E
x=y?    ; <=    A
GTO M   ; <=     R these lines when checking for just two different digits
RTN     ; we get to this point when a fourth digit is reached => next number
LBL M   ; CK=102A 010.5
```

```
DSE I
GTO J   ; go to the next digit to examine
RCL N   ; all digits examined, so now we have an answer
x²      ; proudly show the result
STOP    ; and allow the user to see it
RTN     ; continue with next number



total length:     100.0


;        storage registers
;        I      loop counter for ten digits
;        N      number to examine
;        R      number² being split into digits
;        X Y (Z) the digit to occur
;
;        labels
;        A      loop through all N values
;        B      inner loop
;        C      check max diff digits in square of number N
;        J      place holder
;        M      increment to the next digit to examine
;
;        flags
;        0      set => show progression



regards,
   Bram
```

Re: Short & Sweet Math Challenges #8: Squares !

*Message #19 Posted by **Olivier De Smet** on 22 Apr 2005, 9:51 a.m.,*
in response to message #1 by Valentin Albillo

Here is a quick hack for an HP86B (with advance prog module)

```
10 l=0 @ o=3 @ t=TIME  @ FOR n=33333 TO 99999 @ IF o=10 THEN o=0 @ GOTO 60
20 k=1 @ SFLAG "" @ m=n*n @ SFLAG m MOD 10 @ m=m\10 @ FOR j=2 TO 10 @ b=m MOD 10 @ IF b=0 THEN 60
30 IF FLAG (b) THEN 50
40 IF k=3 THEN 60 ELSE SFLAG b @ k=k+1
50 m=m\10 @ NEXT j @ l=l+1 @ DISP l,n,n*n
60 o=o+1 @ NEXT n @ DISP "Fini:";TIME -t @ END


run
 1                   33334               1111155556
 2                   33335               1111222225
 3                   33338               1111422244
 4                   33359               1112822881
 5                   33989               1155252121
 6                   34641               1199998881
```

```
 7                  34827               1212919929
 8                  34965               1222551225
 9                  36361               1322122321
 10                 37962               1441113444
 11                 39388               1551414544
 12                 39581               1566655561
 13                 40204               1616361616
 14                 43923               1929229929
 15                 44623               1991212129
 16                 47162               2224254244
 17                 50235               2523555225
 18                 54123               2929299129
 19                 54335               2952292225
 20                 57665               3325252225
 21                 58167               3383399889
 22                 62156               3863368336
 23                 66515               4424245225
 24                 66665               4444222225
 25                 66667               4444488889
 26                 66668               4444622224
 27                 67485               4554225225
 28                 68187               4649466969
 29                 68962               4755757444
 30                 70107               4914991449
 31                 70173               4924249929
 32                 72285               5225121225
 33                 72335               5232352225
 34                 72475               5252625625
 35                 76478               5848884484
 36                 78196               6114614416
 37                 78541               6168688681
 38                 78881               6222212161
 39                 79162               6266622244
 40                 80408               6465446464
 41                 81346               6617171716
 42                 81404               6626611216
 43                 81619               6661661161
 44                 81813               6693366969
 45                 83666               6999999556
 46                 86478               7478444484
 47                 97773               9559559529
 48                 98336               9669968896
Fini: 108.019
```

In fact as 0 isn't allowed only integer from 33333 to 99999 are tested and not ending with 0.

The given program is for the second problem (which include the first). Need 1h50 on real hardware to find all solutions. Change line 40 with '40 IF k=2 ...' for the first problem.

Thanks again for the challenge.

Olivier

Edited: 22 Apr 2005, 9:57 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

*Message #20 Posted by **Olivier De Smet** on 22 Apr 2005, 11:16 a.m.,*
in response to message #19 by Olivier De Smet

A better solution for the first problem:

```
list
10 t=TIME
20 FOR i=0 TO 511 @ a=1000000000+VAL (DTB$ (i)) @ b=VAL (DTB$ (511-i))
30 FOR j=1 TO 9 @ FOR k=1 TO 9 @ m=j*a+k*b @ IF FP (SQR (m))=0 THEN DISP SQR (m),m
40 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 517068

run
 81619                6661661161
Fini: 19.118
```

Just 20 mins for finding the value, but need the I/O module.

Olivier

Re: Short & Sweet Math Challenges #8: Squares !

*Message #21 Posted by **J-F Garnier** on 22 Apr 2005, 11:26 a.m.,*
in response to message #20 by Olivier De Smet

Hi Olivier,

I was just about to post a similar solution for the 71B. You can improve this solution by noticing that only numbers ending by 1, 4, 5, 6 or 9 can be a square number. But I didn't find how to solve the 3-digit problem with this approach.

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Message #22 Posted by **Olivier De Smet** on 22 Apr 2005, 11:35 a.m.,*
in response to message #21 by J-F Garnier

Yes I know but searching to eliminate such non solution is too costly in time :)

A better one :

```
list
10 t=TIME
20 FOR i=0 TO 511 @ a=1000000000+VAL (DTB$ (i)) @ b=VAL (DTB$ (511-i)) @ FOR j=1 TO 9 @ m=j*a @ FOR k=1 TO 9 @ m=m+b @ IF FP (SQR (m))=0 THEN DISP SQR (m),m
30 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 517065
run
 81619                6661661161
Fini: 15.199
```

Only 16 mins :)

Olivier

Re: Short & Sweet Math Challenges #8: Squares !

*Message #23 Posted by **J-F Garnier** on 24 Apr 2005, 11:00 a.m.,*
in response to message #22 by Olivier De Smet

A slighly improved version of Olivier's solution, for the HP-71B and using the BSTR$ function from the Math module:

```
10 T=TIME
15 OPTION BASE 1 @ DIM F(5)
16 DATA 1,4,5,6,9
17 READ F()
20 FOR I=0 TO 511 @ A=VAL(BSTR$(I,2))*10+1 @ B=VAL(BSTR$(511-I,2))*10
30 FOR J=1 TO 5 @ M=F(J)*A @ FOR K=1 TO 9 @ M=M+K*B @ IF FP(SQRT(M))=0 THEN DISP SQRT(M),M
40 NEXT K @ NEXT J @ NEXT I @ DISP "Fini:";TIME-T @ END
 81619                6661661161
Fini: 45.77
```

Running on Emu71 using a 1.7GHz processor.

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Message #24 Posted by **Olivier De Smet** on 25 Apr 2005, 3:02 a.m.,*
in response to message #23 by J-F Garnier

Hi,

As I told you I don't need to optimize, my programm need only 15.2 sec running on an HP86B emulator on a 1.8Ghz pentium M :)

But you're right it's faster like this :

```
list
10 t=TIME  @ OPTION BASE 0@ DIM f(4)
20 f(0)=1 @ f(1)=4 @ f(2)=5 @ f(3)=6 @ f(4)=9
30 FOR i=0 TO 511 @ a=VAL (DTB$ (i))*10+1 @ b=VAL (DTB$ (511-i))*10 @ FOR j=0 TO 4 @ m=a*f(j) @ FOR k=1 TO 9 @ m=m+b @ IF FP (SQR (m))=0 THEN DISP SQR (m),m
40 NEXT k @ NEXT j @ NEXT i @ DISP "Fini:";TIME -t @ END
 516894
run
 81619                6661661161
Fini: 8.486
```

Only 8.846 sec :)

Olivier

Edited: 25 Apr 2005, 6:02 a.m.

Re: Short & Sweet Math Challenges #8: Squares !

*Message #25 Posted by **Olivier De Smet** on 22 Apr 2005, 11:39 a.m.,*
in response to message #21 by J-F Garnier

For 3 digit as someone said it there is too much candidates, so the reverse approach should be better I think because you only have to test (99999-33333) candidates.

For 2 digits it's the cost of the test which is important as you have either 66666 or 512*9*9 candidates.

Olivier

Hi, Jean-François !

*Message #26 Posted by **Valentin Albillo** on 22 Apr 2005, 11:46 a.m.,*
in response to message #21 by J-F Garnier

Hi, J-F:

Your wonderful Emu71 HP-71B's emulator runs my solution program for the HP-71B in under 40 seconds !

By the way, did you get to see my latest HP-71B articles published in Datafile ? All of the featured programs were created with the help of your Emu71, and it runs them like a charm. I do include footnotes recommending both your emulator and Hrastprogrammer's one for the 48/49, so that readers can know that they can try the programs and fully enjoy them themselves.

The ones featured in the latest Datafile issue are an extremely good test of your emulator's performance, as they make use not only of the emulated basic 71B itself, but also of the emulated Math and HP-IL ROMs as well, and it does certainly pass the taxing ordeal with flying colors, to say the least !

Best regards from V.

Re: Hi, Valentin

*Message #27 Posted by **J-F Garnier** on 24 Apr 2005, 9:52 a.m.,*
in response to message #26 by Valentin Albillo

>> By the way, did you get to see my latest HP-71B articles published in Datafile ?

Unfortunatly no, I (still) didn't subscribe to Datafile, shame on me ...

J-F

Re: Short & Sweet Math Challenges #8: Squares !

*Message #28 Posted by **Marcus von Cube** on 22 Apr 2005, 1:36 p.m.,*
in response to message #1 by Valentin Albillo

This is a solution for the HP 42S and HP 41C. [Edited]

Usage:

Enter the number of allowed different digits (2 or 3) in X and XEQ "SQ10". After the 33333 is displayed you have to press R/S (You can correct the starting number, if you like.)

If you enter a number >9, XEQ "SQ10" will just run the comparison engine (LBL 00) and show "OK" if the criterion is met. You need to store the number of digits manually in R02 before you can use the program this way.

```
LBL "SQ10"   Main entry point
0
STO 09
Rv
9
X<>Y
X<=Y?
GTO 10
XEQ 00
RCL 00
FC? 00
RTN
"OK"
AVIEW
RTN
```

Now comes the checking engine. It works on flags 0-9 which are all set in the beginning and cleared indirectly for each corresponding digit. The reset operations are counted in R00. If the count reaches the limit in R02, the check has failed and flag 0 will be cleared. (This is done at no cost, if a "0" digit is found.) A cleared flag 0 stops the loop.

R09 counts the total number of calls for statistical purposes.

```
LBL 00
VIEW ST X
SF 00
SF 01
SF 02
SF 03
SF 04
SF 05
SF 06
SF 07
SF 08
SF 09
0
STO 00
1
STO+ 09
Rv                RDN for HP 41
Rv


LBL 01            digit loop
STO 01
10
MOD
FC?C IND ST X     this does the flag testing and clearing!
GTO 02            this digit is already counted
RCL 00            get count
RCL 02            max # of different digits allowed
X<=Y?
```

```
CF 00
FC? 00
RTN               too many different digits
1
STO+ 00           count
LBL 02            get next digit
RCL 01
10
/
IP                INT for HP 41
X!=0?             any digits left?
GTO 01
RTN


LBL 05            find smallest digit from flags
                  works only after a passed check!
RCL 00
RCL 02
-
X=0?              maximum # of digits reached?
GTO 06            look for smallest
Rv
1                 return 1, because less than the
RTN               allowed number of digits were present


LBL 06
1
+
FS? IND ST X
GTO 06
RTN
```

This is the controlling loop. It starts at 33333+1 but it does not try each and every number up to 99999. When the left most 4 digits do not pass the check, there is no need to try any number that would result in the same 4 digits. I simply construct a number with the next higher value and get the SQRT from it:

If guess² == p,qrs,tuv,wxy. then new guess = SQRT((p,qrs.+X)*10e6 + Y11,111.)

p,qrs+X is determined by adding 1 in a loop and checking the resulting 4 digit number until the check passes. Y is the smallest digit found in p,qrs+X by the check routine.

A similar trick is used to skip guesses when the first 3 digits fail the check.

(The following is awful spaghetti code and I'm willing to streamline it when I have time.)

```
LBL 10
STO 02            store # of allowed digits
0
STO 05
STO 06            scratch registers for first 3 to 4 digits
33333             the loop starts with this initial guess
STOP              correct the initial guess at will
                  (The loop starts with this number + 1!)


LBL 11
```

```
STO 03          store first guess - 1


LBL 12
1
STO+ 03         increment guess
999999          final value
RCL 03
X>Y?
GTO 20          no more numbers to try
X^2
STO 04          store result to check (for later)
1E6
/
IP              first 4 digits (INT for HP 41)
x<>Y
RCL 05          first 4 digits from last loop
X!=Y?           changed?
GTO 13          check and find best next guess
RCL 04          squared guess
XEQ 00          check it
FC? 00
GTO 12          failed, try next
CLD             OK, show result
RCL 03
RCL 04
BEEP
STOP
GTO 12          back to main loop


LBL 13          find the next guess quicker by checking
1               the first 3 or 4 digits only
-
STO 05


LBL 14
1
STO+ 05         new value for first 4 digits
RCL 06          first 3 digits (saved)
RCL 05          first 4
10
/
IP
X=Y?
GTO 19          the first 3 are the same as before


LBL 15
STO 06          new value for first 3
XEQ 00          check
FS? 00
GTO 16          first 3 digits are OK, must try first 4
RCL 06
1
+
GTO 15
```

```
LBL 16            new first 4 digits from first 3 digits
RCL 06
10
*
XEQ 05            find smallest digit from flags
+
STO 05


LBL 19            check the first 4 digits
RCL 05
XEQ 00            check
FC? 00
GTO 14            check failed, try next
RCL 05            compute new guess
10
x
XEQ 05            find smallest digit
+
1E5
x
11110
+
SQRT
IP
GTO 11            restart with new guess


LBL 20            finis
CLD
CLX
END
```

See next post for some statistics.

Edited: 23 Apr 2005, 2:33 p.m. after one or more responses were posted

Some Statistics for the above

*Message #29 Posted by **Marcus von Cube** on 23 Apr 2005, 11:09 a.m.,*
in response to message #28 by Marcus von Cube

Performance:

The 42S took about 90 minutes to find the 2 digit solution. My 41CX arrived after 4 hours and 10 minutes.

The number of checks performed was 4487 until the first solution (81619) and 6625 total. (These numbers were computed before I've made my last changes, but the results should be similar.)

Anybody for a faster check routine? It looks like the loop over the digits takes most of the computing time.

Re: Some Statistics for the above

*Message #30 Posted by **Marcus von Cube** on 24 Apr 2005, 8:28 a.m.,*

in response to message #29 by Marcus von Cube

With the current version I could reduce the time and number of calls further:

3771 calls to the check routine and 75 minutes to solve the 2 digit problem on a real unmodified 42S. The 41C took 3:45 minutes. (It's funny to watch the flags toggle in the display :-))

The total number of calls to the checking routine was 5140. This will be significantly higher for the three digit problem, because less numbers can be ruled out beforehand based on the first 4 digits.

Re: Short & Sweet Math Challenges #8: Squares !

*Message #31 Posted by **Arnaud Amiel** on 23 Apr 2005, 6:47 p.m.,*
in response to message #1 by Valentin Albillo

As stated above, I believed that assembly was more suited to this problem so I gave it a try. The 49G and 49g+ are the only calcs to have assembly available out of the box so I tried on the 49g+. The program below (non optimised) solves question 2 in 43s. A 49g would take 1min 43s. It is easy to port it on the 48 where it should run as fast as on a 49 (or half speed on the SX).

Here is the program

```
SAVE
SETDEC
LC 33332
R1=C.A


*NextNumber
C=R1.A C+1.A SKIPNC { P=0 SETHEX LOADRPL }
R1=C.A
C=0.W C=R1.A
CSL.W CSL.W CSL.W
A=C.W
GOSBVL =SPLITA
C=B.W D=C.W C=A.W
GOSBVL =MULTF
GOSBVL =PACK
C=0.W C=A.M
CSR.W CSR.W CSR.W CSR.W CSR.W


P=10 A=0.S B=0.S D=0.S
*TestNumber
P-1 GOC GoodNumber
CSRC
?C=0.S GOYES NextNumber
?A#0.S { A=C.S GOTO TestNumber }
?A=C.S GOYES TestNumber
?B#0.S { B=C.S GOTO TestNumber }
?B=C.S GOYES TestNumber
?D#0.S { D=C.S GOTO TestNumber }  %remove for question 1
?D=C.S GOYES TestNumber           %remove for question 1
GOTO NextNumber


*GoodNumber
```

```
A=0.W A=R1.A ASRC ASRC ASRC ASRC ASRC ASRC  A+4.A
GOSBVL =PUSH%
SAVE
SETDEC
GOTO NextNumber
```

As usual archive before playing with assembly

Arnaud

PS: If you want I can send you the compiled program for 48 or 49 by email

[LONG] S&SMC #8: My original solutions, comments and curiosities !

*Message #32 Posted by **Valentin Albillo** on 25 Apr 2005, 5:13 a.m.,*
in response to message #1 by Valentin Albillo

Hi all,

Thanks to all of you interested in my S&SMC #8 and most specially to those who provided such ingenuous and elegant solutions for such a variety of HP calcs and computers, we really saw a lot of interesting techniques being used here for the one and only purpose: find the solutions and find them fast. Congratulations to all of you, I *really* feel most satisfied with your contributions !

As promised, this is my original, 4-line solution for the HP-71B:

```
10 C=0 @ FOR N=33334 TO 99999 @ M$=STR$(N*N) @ IF POS(M$,"0") THEN 40 ELSE T$=M$[1,1]
20 FOR I=2 TO 10 @ IF POS(T$,M$[I,I]) THEN 30 ELSE IF LEN(T$)=3 THEN 40 ELSE T$=T$&M$[I,I]
30 NEXT I @ C=C+1 @ DISP N;M$,T$
40 NEXT N @ DISP "OK:";C;"found"
```

Calling it produces the 48 solutions, displaying the number, its square, and the distinct digits it consists of:

```
>CALL


 33334 1111155556    156
 33335 1111222225    125
 33338 1111422244    142
 33359 1112822881    128
 33989 1155252121    152
 34641 1199998881    198
         ...
 81404 6626611216    621
 81619 6661661161    61    <- only *two* different digits, the solution to the first part
 81813 6693366969    693
 83666 6999999556    695
 86478 7478444484    748
 97773 9559559529    952
 98336 9669968896    968


OK:  48  found
```

It takes 3 min 28 sec when run under Emu71 in a 1.4 Ghz PC. The program searches all numbers which generate 10-digit squares fulfilling the conditions. As no 0's are allowed, the smaller possible square would be 1111111111, so we begin the search at SQR(1111111111) rounded to the nearest integer, which is 33334, and go on searching till SQR(9999999999), which comes out to 99999. We

square each number in turn and convert the result to a string. The first POS then immediately discards numbers having a "0" anywhere. Then we keep on extracting individual digits which are added to an auxiliary string, using POS to detect repetitions (in which case we don't add the digit again) and LEN to see if there are more then 3 distinct digits already collected, in which case we forfeit further processing of this number and go instead to check the next candidate.

However, this is not the fastest way to proceed because the HP-71B is optimized for numerical computations and string processing is relatively slow. So I originally came out instead with this other version, which uses flags to register which digits have been used:

```
10 C=0 @ FOR N=33334 TO 99999 @ M=N*N @ CFLAG ALL @ P=0
20 FOR I=1 TO 10 @ D=MOD(M,10) @ IF NOT D THEN 50 ELSE M=M DIV 10
30 IF FLAG(D) THEN 40 ELSE IF P=3 THEN 50 ELSE SFLAG D @ P=P+1
40 NEXT I @ C=C+1 @ DISP N;N*N
50 NEXT N @ DISP "OK:";C;"found"
```

It works the same, only using flags instead of strings. Each candidate number is decomposed into its constituent digits. If the digit is a 0, we skip to the next candidate, else a flag is set corresponding to the digit. If more than 3 flags have alrady been set, no further digits are isolated but we go instead to test the next candidate. This version finds the exact same solutions but it takes just 72 seconds (under Emu71), i.e., it's 3 times faster than the string version. A real HP-71B takes much longer of course, but it's a real beauty to see the display's flag indicators making a frantic dance while the program runs !

In both versions, changing the "3" to any other single digit N (1-9) would find squares made up of N distinct digits. For instance, with N=2, it produces 6661661161, the only 10-digit square consisting of just two different digits, 1 and 6 in this case. Also, you can search for 12-digit squares instead, say, by simply changing the search limits to 333334 and 999999 respectively, and by changing the limit of the I loop to 12. This can be automated by simply creating a new, generalized version that asks the user for the number of digits and maximum different values, like this:

```
 1 INPUT "# Digits (1-12) = ";U @ INPUT "# Diff. val. (1-9) = ";V
10 C=0 @ FOR N=INT(SQR((10^U-1)/9)) TO INT(SQR(10^U-1)) @ M=N*N @ CFLAG ALL
20 P=0 @ FOR I=1 TO U @ D=MOD(M,10) @ IF NOT D THEN 50 ELSE M=M DIV 10
30 IF FLAG(D) THEN 40 ELSE IF P=V THEN 50 ELSE SFLAG D @ P=P+1
40 NEXT I @ C=C+1 @ DISP N;N*N
50 NEXT N @ DISP "OK:";C;"found"
```

Let's try it ! To find the five 5-digit squares consisting of at most 2 different values:

```
>RUN


# Digits (1-12)    = 5  [ENTER]
# Diff. val. (1-9) = 2  [ENTER]


 109  11881
 173  29929
 212  44944
 235  55225
 264  69696


OK: 5 found
```

While to find the thirty 12-digit squares consisting of at most 3 different values:

```
>RUN


# Digits (1-12)    = 12 [ENTER]
# Diff. val. (1-9) = 3  [ENTER]
```

```
 333334  111111555556
 333335  111112222225
 333338  111114222244
 333359  111128222881
 333858  111461164164
 363639  132233322321
 387288  149991994944
 461761  213223221121
 492162  242223434244
 505525  255555525625
 515415  265652622225
 586893  344443393449
 649788  422224444944
 658357  433433939449
 664388  441411414544
 665908  443433464464
 666515  444242245225
 666665  444442222225
 666667  444444888889
 666668  444446222224
 682908  466363336464
 782104  611686666816
 805262  648446888644
 818333  669668898889
 828667  686688996889
 834386  696199996996
 869924  756767765776
 939938  883483443844
 974417  949488489889
 994836  989698666896


OK: 30 found
```

The 'flags' version also has the added advantage that it translates very easily to RPN machines with little or no alpha capabilities, such as the HP-41C or HP-15C. For instance, here's the translated version for an HP-41CX, a meager 46 steps:

```
01 *LBL "SQ3"   16  STO 04      31  GTO 04
02  CLX         17   10         32 *LBL 05
03  X<>F        18  STO 05      33   1
04  CF 08       19 *LBL 01      34  ST+ 02
05  CF 09       20  RCL 03      35   1E5
06  RCLFLAG     21  INT         36  RCL 02
07  STO 01      22   10         37  X#Y?
08  33334       23  ST/ 03      38  GTO 00
09  STO 02      24  MOD         39  STOP
10 *LBL 00      25  X=0?        40 *LBL 04
11  X^2         26  GTO 05      41  DSE 05
12  STO 03      27  FS? IND X   42  GTO 01
13  RCL 01      28  GTO 04      43  RCL 02
14  STOFLAG     29  SF IND X    44  X^2
15  .003        30  ISG 04      45  VIEW X
                                46  GTO 05
```

It runs under the V41 emulator in 72 min (a real HP-41CX would take a lot longer) to produce all 48 solutions, and again, it's a joy to see all those flag indicators frantically turning on and off:

```
[R/S]


 1111155556
 1111222225
    ...
 9559559529
 9669968896
```

For practical reasons and in order to to make it accessible to most HP calculators, this challenge specifically addressed 10-digit squares, but if you feel like it there are marvels out there waiting to be discovered, here are some examples:

```
    100995109391549797 51^2 = 10200012121011110110212001110122002 2001 (39 digits, all 0,1,2)


   5579635589546259268 61^2 = 31132333312131232233213332311122332 1313321 (42 digits, all 1,2,3)


675754811056988742949784^2 = 456644564666666555445565455644644555565545646656 (48 digits, all 4,5,6)
```

Actually, even the 48-item list of solutions for the original challenge holds a number of very interesting results upon close inspection, such as these beautifully-patterned squares:

```
  1111155556  ( 11111-5555-6    )
  1111222225  ( 1111-22222-5    )
  1616361616  ( 16-16-36-16-16  )
  2929299129  ( 29-29-29-91-29  )
  4444222225  ( 4444-22222-5    )
  4444488889  ( 44444-8888-9    )
  5252625625  ( 52-52-625-625   )
  6617171716  ( 66-17-17-17-16  )
```

Thanks for your interest in this humble S&SMC #8, hope you enjoyed it, and

Best regards from V.

Re: Short & Sweet Math Challenges #8: Squares !

*Message #33 Posted by **Tizedes Csaba [Hungary]** on 27 Apr 2005, 7:33 a.m.,*
in response to message #1 by Valentin Albillo

:) Err, I missed it...! Thanks for the challenge! Csaba

You're welcome, Csaba ! [N.T.]

*Message #34 Posted by **Valentin Albillo** on 27 Apr 2005, 9:30 a.m.,*
in response to message #33 by Tizedes Csaba [Hungary]

Best regards from V.

[Return to Index | Top of Index]