HP Forum Archive 17

A modest arithmetical challenge

Message #1 Posted by ***Karl Schneider*** *on 14 Apr 2007, 8:58 p.m.*

Hi, all --

Here is an interesting arithmetical problem that appeared in a student-engineer magazine in 1993 or 1994:

Assume that each integer from 1 through 9 is used *exactly once* to form three quotients of single-digit integer numerators and two-digit integer denominators. How many ways could the digits be arranged such that the sum of these three terms equaled unity?

Example:

```
  7      8      9     181
---- + ---- + ---- =  ---  ~= 1.005
 12     36     45     180
```

is almost a solution.

I obtained the above result by trial-and-error, but was unable to analytically deduce the correct answer. In 1995, I wrote a recursive C-language program that found the solution by "brute force", evaluating all 9! = 362,880 possible combinations of digits. Of course, there are six ways to arrange the three terms, so there are "only" 60,480 *unique* combinations.

Several years ago, I showed this problem to a colleague who offered the following:

```
"There is a subset that populates a more general relationship,


 A     C     E
--- + --- + --- =  1
 B     D     F


where A is odd and 2*D = F.
```

```
Example:  1/2 + 1/3 + 1/6 = 1."
```

He obtained the correct solution by trial and error within this subset of combinations. I can't offer a proof that the stated equation is generally true, or why it may have been a necessary (not merely sufficient) condition for the solution.

So, the challenge: Can anyone provide a direct analytical solution to this problem? If not, would anyone dare to attempt to write an efficient HP calculator program to solve the problem?

I believe that the HP-15C offers sufficient programming features to tackle the problem using the brute-force method:

- Enough storage registers to maintain nine separate digits and to store solutions
- Nine usable flags (flags 0-9 excluding flag 8) to track whether a digit is "in use"
- Subroutines, conditional tests, and plenty of labels
- A versatile indirect register

So, a workable HP-15C program could be written. However, I also believe that such a program might take literally weeks to run to completion on an actual HP-15C! Recursion and integer arithmetic could not be employed, as my C program does.

The HP-71B would be more suitable, with 12 times the speed, BASIC language, and recursion.

Of course, with computing speed being an important issue, a PC-based simulator would be a better choice than an actual calculator.

I doubt that there is any trivially-simple derivation of the solution; I also believe that an efficient program would require time and effort. Still, the mathematical knowledge and programming skills offered in response to Valentin's challenges never ceases to impress me. I'll post my C-language program shortly, as a guide for those who might find it useful.

Best regards,

-- KS

Re: A modest arithmetical challenge

*Message #2 Posted by **Valentin Albillo** on 14 Apr 2007, 10:49 p.m.,*
in response to message #1 by Karl Schneider

Hi, Karl:

Nice arithmetic challenge. Regrettably, I don't have the time to produce an efficient and elegant solution which only tests unique permutations of the 9 digits and uses integer arithmetic and recursion as your C program does, nor can I indulge in trying bitmasks and boolean operations to discard impossible combinations if using another approach not requiring the recursive generation of permutations, but this extremely quick-and-dirty, 14-line brute-force HP-71B program which took me only about 10 minutes to write, enter, and run, actually does the job:

```
 1 DESTROY ALL @ FOR A=1 TO 9 @ FOR D=A+1 TO 9 @ FOR G=D+1 TO 9
 2 FOR E=1 TO 9 @ IF E=A OR E=D OR E=G THEN 13
 3 FOR F=1 TO 9 @ IF F=A OR F=D OR F=G OR F=E THEN 12 ELSE N=10*E+F
 4 FOR B=1 TO 9 @ IF B=A OR B=D OR B=G OR B=E OR B=F THEN 11
 5 FOR C=1 TO 9 @ IF C=A OR C=D OR C=G OR C=E OR C=F OR C=B THEN 10
 6 M=10*B+C @ R=M*N @ S=A*N+D*M
 7 FOR H=1 TO 9 @ IF H=A OR H=D OR H=G OR H=E OR H=F OR H=B OR H=C THEN 9
 8 P=9*H+45-A-B-C-D-E-F-G @ IF R*(P-G)=P*S THEN DISP A;"/";M;"+";D;"/";N;"+";G;"/";P;"= 1"
 9 NEXT H
10 NEXT C
11 NEXT B
12 NEXT F
13 NEXT E
14 NEXT G @ NEXT D @ NEXT A @ DISP "OK"


>RUN


   5/34 + 7/68 + 9/12 = 1


   OK
```

and it finds the unique solution (and the fact that there are no others) in about one minute under Emu71, which translates to about 4 hours in a physical HP-71B. Improvements like the ones mentioned above would easily diminish these times enormously, but it would take me more than 10 minutes in all to concoct. :-)

Thanks for your interesting and entertaining challenge and

Best regards from V.

Re: A modest arithmetical challenge

Message #3 Posted by ***Karl Schneider*** *on 15 Apr 2007, 3:54 p.m.,*
in response to message #2 by Valentin Albillo

Hi, Valentin --

Well, I can't verify your astounding claim of only 10 minutes for analysis, development and entry of code, and testing. Still, I'm quite impressed by your analytical and programming skills. Your program did indeed produce the only correct result.

A solution based on Fortran 77 (which does not support dynamic memory allocation for recursion), would likely utilize eight or nine nested loops, as you did using BASIC. I developed my C program for learning experience after my collegiate course in C showed that recursion was supported. Recursion can make the code more compact and flexible, but likely slows the execution due to O/S overhead for assignment and release of RAM.

I will likely post my C program on Monday evening (Tuesday morning for those of you in Europe), because an electronic copy of the source code is not readily available at home.

Your BASIC program as it is structured -- which is not difficult to interpret -- may not port readily to a "clean" RPN-keystroke program absent excessive conditional tests and GTO's. However, despite your description of it as "quick-and-dirty", I see clear evidence of logical structure and thought. Furthermore, it already performs the optimization not to execute redundant permuatations of digits by setting in line 1 the condition A < D < G (which are the three numerators). Your program also wisely calculates the common-denominator products instead of the three quotients. I had done the same in my C program both to eliminate the possiblity of roundoff errors and to exploit the much-faster integer arithmetic.

A program that I would write for an HP-15C/32S/32SII/33S/41*/42S would probably emulate the approach I used for my C program, which uses a variable in the fashion of a Fortran LOGICAL variable to identify which digits were already "in use". Flags would fill that role on an HP calculator. Use of an indirect register would reduce redundant code considerably.

Thanks again for your informative contribution,

-- KS

Edited: 15 Apr 2007, 5:00 p.m.

Re: A modest arithmetical challenge

Message #4 Posted by ***Valentin Albillo*** *on 15 Apr 2007, 7:09 p.m.,*
in response to message #3 by Karl Schneider

Hi again, Karl:

Karl posted:

> *"Well, I can't verify your astounding claim of only 10 minutes for analysis, development and entry of code, and testing. Still, I'm quite impressed by your analytical and programming skills. Your program did indeed produce the only correct result."*

Thanks a lot for your appreciation but neither my 10-min claim is astounding nor does this "klutzy" piece of code demonstrate much about my "skills". :-)

The 10-min claim is easily justified by simply looking at my program: it is nothing but a series of brute-force loops, with the only proviso being the storage of invariant intermediate values out of the loops which doesn't affect their value, the final digit being directly computed as 45 less the sum of the others (as all 9 digits add up to 45), and clumsily testing each digit against the previous ones to make sure no digits are repeated.

This straightforward logic can be keyed in very quickly, as the HP-71B lets you enter a variant of a program line by simply editing its line number, kind of a copy-paste operation. You just enter first the most complicated FOR ... IF ... line, then edit its variable and line number (deleting the final test) to enter the next simplest one, which is then edited likewise to produce the next one, etc. (BTW, anyone who's got Emu71 can enter this code by simply doing a real copy-paste of the code from the posted message to Emu71, no need to actually key anything in).

As for testing time it's actually zero because, matter of fact, everything is so simple that no debugging was needed, it produced the correct solution on its 'maiden' run.

"I will likely post my C program on Monday evening (Tuesday morning for those of you in Europe)"

It will be interesting to have a look at it. A recursive solution for the HP-71B can be concocted in much fewer lines than my posted iterative one, so your C code must also be extremely compact.

"However, despite your description of it as "quick-and-dirty", I see clear evidence of logical structure and thought."

It was all trivial, Karl, it's only that I didn't have the time to have a go at an elegant, efficient solution as you originally asked for, but not wanting to let your challenge unanswered on my part, I did what I could in 10 min. If you check the timestamp of my posted solution, you'll see that I saw it and wrote a quick'n'dirty solution for it by 10:49 P.M. MoHP Forum's Official Time, which translates as 4:49 A.M. Spanish Time, i.e., 4:49 in the midst of the night, after a very hard daytime with some certification training, with a mild headache and practically asleep.

If I get the time, I'll try to post a decent solution but I doubt I will be able to do that before you (or someone else) post your own. In the meantime, some trivial modifications to my posted program produce these interesting additional results:

```
Largest sum    :  7/46 + 8/25 + 9/13 = 1.16448160535    (17409/14950)


Smallest sum   :  1/74 + 2/85 + 3/96 = 0.0682929252782  (6873/100640)


Closest sum >1 :  4/76 + 8/23 + 9/15 = 1.00045766590    (2186/2185)
```

```
    Closest sum <1 :  5/87 + 6/24 + 9/13 = 0.999778956676   (4523/4524)
```

On the analytic front, it's easy to prove that in order for the three fractions to be able to add up to 1 or greater, the digit 1 must forcibly be the first digit of a denominator, and the corresponding numerator must then be > 5. Making use of this shortcut will help save lots of running time.

Thanks for your delightful arithmetic challenge, seconds please ! :-)

Best regards from V.

Edited: 16 Apr 2007, 9:07 a.m.

Discussion ("A modest arithmetical challenge")

Message #5 Posted by ***Karl Schneider*** *on 19 Apr 2007, 12:42 a.m.,*
in response to message #4 by Valentin Albillo

Hi, Valentin --

Quote:

> Thanks a lot for your appreciation but neither my 10-min claim is astounding nor does this "klutzy" piece of code demonstrate much about my "skills". :-)

Oh, I wouldn't denigrate your own work, even if it doesn't represent your finest efforts. Hardly anyone -- including myself -- could have immediately written a working program to solve the problem without a fair amount of forethought.

Quote:

> It was all trivial, Karl, it's only that I didn't have the time to have a go at an elegant, efficient solution as you originally asked for, but not wanting to let your challenge unanswered on my part, I did what I could in 10 min.

As far as I'm concerned, your effort surpassed the specification. I never stated "elegant", and "efficient" implied only that it would run in a reasonable amount of time without exhausting multiple sets of batteries. The techniques of computational reduction and the "special sums" you offered were an added bonus.

Quote:

> Thanks for your delightful arithmetic challenge, seconds please ! :-)

Sorry, but I don't have any others in store at the moment. I suspect, though, that any I had would cause me "Mom's lament": An hour to prepare, and ten minutes to demolish... :-)

Best regards,

-- KS

Edited: 19 Apr 2007, 12:44 a.m.

Re: Discussion ("A modest arithmetical challenge")

Message #6 Posted by ***Gerson W. Barbosa*** *on 19 Apr 2007, 8:25 p.m.,*
in response to message #5 by Karl Schneider

> Quote:
>
> I suspect, though, that any I had would cause me "Mom's lament": An hour to prepare, and ten minutes to demolish... :-)

Karl,

Don't worry about it being solved that soon. Ten minutes really appears to be the standard time to write a program to solve your challenge:

Quoted from http://users.skynet.be/worldofnumbers/ninedig3.htm:

> Quote:
>
> The solution is very easy to search for, Preon tells. It is
>
> ```
> (9/12) + (7/68) + (5/34) = 1
> ```
>
> *"The solution is unique as my program revealed, took maybe 10 minutes*
>
> *to write the code and 1 minute to run and display the only answer."*

Having it take me at least 100 minutes to write that ugly piece of Turbo Pascal code only shows I was right not having tried to pursue a career in computer programming :-)

Best regards,

Gerson.

Edited: 19 Apr 2007, 8:37 p.m.

Re: A modest arithmetical challenge

Message #7 Posted by ***Gerson W. Barbosa*** *on 14 Apr 2007, 11:30 p.m.,*
in response to message #1 by Karl Schneider

Hello Karl,

> Quote:
>
> ```
> "There is a subset that populates a more general relationship,
>  A     C     E
> --- + --- + --- =  1
>  B     D     F
> where A is odd and 2*D = F.
> Example:  1/2 + 1/3 + 1/6 = 1."
> ```

The relationships below might also be useful to reduce the use of brute force:

```
 A     C         E
--- + --- = 1 - ---
 B     D         F


 A*D + B*C    F - E             E = F - A*D - B*C
---------- = -------      =>
   B*D         F                F = B*D
```

Regards,

Gerson.

Re: A modest arithmetical challenge

Message #8 Posted by ***Gerson W. Barbosa*** *on 15 Apr 2007, 7:10 p.m.,*
in response to message #7 by Gerson W. Barbosa

Not an elegant and concise solution as Valentin's, but this Turbo Pascal 3 program does find the solution:

```
--------------------------------------------------------------
Program KarlsProblem;
var a, b, c, d, e, f: integer;
                   x: real;
                   t: boolean;
begin
  ClrScr;
  a:=-1;
  repeat
    a:=a+2;
    for b:=12 to 49 do
      for c:=1 to 9 do
        for d:=12 to 98 do
          begin
            t:= (a <> b div 10) and (a <> b mod 10) and (a<>c) and (a <> d div 10) and (a <> d mod 10);
            t:= t and ((b div 10)<>(b mod 10)) and ((b div 10)<>c) and ((b div 10)<>(d div 10)) and ((b div 10)<>(d mod 10));
            t:= t and ((b mod 10)<>c) and ((b mod 10)<>(d div 10)) and ((b mod 10)<>(d mod 10));
            t:= t and (c<>(d div 10)) and (c<>(d mod 10));
            t:= t and ((b mod 10)*(d mod 10)<>0);
            if t and (d=2*b) then
              begin
                for e:=1 to 9 do
                  for f:=12 to 98 do
                    begin
                      t:= (a + c + e + (b div 10) + (b mod 10) + (d div 10) + (d mod 10) + (f div 10) + (f mod 10))=45;
                      t:= t and (e<>a) and (e<>c);
                      t:= t and (e<>(b div 10)) and (e<>(b mod 10)) and (e<>(d div 10)) and (e<>(d mod 10));
                      if t and (e<>(f div 10)) and (e<>(f mod 10)) and ((f div 10)<>(f mod 10)) then
                        begin
                          x:=(a/b+c/d+e/f);
                           if (x<1.0000001) and (x>0.9999999) then
                             WriteLn(a:2,'/',b:2,' +',c:2,'/',d:2,' +',e:2,'/',f:2,' = ',x:1:0)
                        end
                    end
              end
            end
  until a=9
end.

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

The output is:

```
 5/34 + 7/68 + 9/12 = 1
```

The tests in the outer loop filter 232 candidates. Additional tests are performed in the innermost loop in order to obtain the final solution. This runs instantaneously on my slow computer but I guess it would take too much time on the HP-71B, even longer on the HP-33S in case it could be ported to it. This is still brute force; I cannot think of a way to speed it up.

Karl's problem is a *pandigital* one, that is, a problem whose solution involves the use of nine or ten unrepeated digits. There seems to be variations: some allow the use of decimal point, square root symbol, factorial and other functions whilst some allow only integers numbers and the arithmetic operators.

Incidentally, I was playing with these when Karl posted his interesting problem:

```
'71/(2*9+4.60)-8^(-sqrt(53))' (to be evaluated on a ten-digit calculator)
```

or

```
'710/CEIL(sqrt(26^4/9))-8^(-sqrt(53))'
```

This looks like cheating, but it's hard to get 226 out of the digits 2, 4, 6 and 9 :-)

Gerson.

Edited: 16 Apr 2007, 6:43 p.m.

Another approach

Message #9 Posted by ***Gerson W. Barbosa*** *on 16 Apr 2007, 10:11 p.m.,*
in response to message #8 by Gerson W. Barbosa

Let's give luck a chance:

```
------------------------------------
10 DEFINT J-N
12 DEFSNG X
15 CLS
17 RANDOMIZE (7)
20 FOR K = 0 TO 8
30   N(K) = K + 1: M(K) = 0
40 NEXT K
34 J = 0
50 K = (RND * 8)
```

```
60 IF N(K) <> 0 THEN M(J) = N(K): N(K) = 0: J = J + 1
70 IF J < 9 THEN 50
80 X = M(0) / (10 * M(1) + M(2)) + M(3) / (10 * M(4) + M(5)) + M(6) / (10 * M(7) + M(8))
90 IF (X < .99999) OR (X > 1.00001) THEN 20
100 PRINT M(0); "/"; 10 * M(1) + M(2); "+"; M(3); "/"; 10 * M(4) + M(5); "+"; M(6); "/"; 10 * M(7) + M(8); "="; X
------------------------------------
```

Output:

```
 5 / 34 + 7 / 68 + 9 / 12 = 1
```

This QBASIC program finds the solution in about 15 seconds (Pentium III @ 500 MHz). The random generator seed in line 17 was chosen arbitrarily.

P.S.: Only 14 lines as I've just noticed, but this took at least 20 minutes between the first idea and the printing of the answer :-)

These two lines should be replaced in the QBASIC program above:

```
30   N(K) = K + 1


45 J = 0
```

By the way, it's easy to port the program to the HP-71B:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ RANDOMIZE 8
20 FOR K=0 TO 8 @ N(K)=K+1 @ NEXT K @ J=0
30 K=INT(RND*9) @ IF N(K)#0 THEN M(J)=N(K) @ N(K)=0 @ J=J+1
40 IF J<9 THEN 30
50 X=M(0)/(10*M(1)+M(2))+M(3)/(10*M(4)+M(5))+M(6)/(10*M(7)+M(8)) @ IF X#1 THEN 20
60 DISP M(0);"/";10*M(1)+M(2);"+";M(3);"/";10*M(4)+M(5);"+";M(6);"/";10*M(7)+M(8);"=";X


>RUN
 5 / 34 + 7 / 68 + 9 / 12 = 1
```

The result was found in 57.75 seconds (Emu71 @500MHz), 1 h 40 min 53 sec on the real HP-71B, according to the built-in timer. Previoulsy I had run the program on the emulator once with the original seed but I quit after one hour with no result. This was the second seed I tried.

Some interesting seeds:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ RANDOMIZE 2.9


10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ RANDOMIZE 10.99


10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ RANDOMIZE 82.622


10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ RANDOMIZE 82.583
```

Respectively 71.9, 51.8, 12.56 and 1.43 seconds on the **real** HP-71B. This is plain cheating though :-)

The equivalent Turbo Pascal program finds the result in 3 minutes and 20 seconds on the HP-200LX, the first time it is run:

```
Program Karls_Problem;
  var a, b: array[1..9] of byte;
       i,j: byte;
         r: real;
begin
  ClrScr;
  repeat
    for i:=1 to 9 do
      a[i]:=i;
    i:=1;
    repeat
      j:=Random(9)+1;
      if a[j] <> 0 then
        begin
          b[i]:=a[j]; a[j]:=0;
          i:=i+1
        end
    until i=10;
    r:=b[1]/(10*b[2]+b[3])+b[4]/(10*b[5]+b[6])+b[7]/(10*b[8]+b[9])
  until (r>0.99999) and (r<1.00001);
  for i:=0 to 2 do
    Write(b[3*i+1]:1,'/',b[3*i+2]:0,b[3*i+3],'+');
  WriteLn(#8,'=',r:1:0)
end.
```

```
9/12+7/68+5/34=1
```

Edited: 28 Apr 2007, 10:57 p.m.

Re: A modest arithmetical challenge

Message #10 Posted by ***allen*** *on 15 Apr 2007, 3:41 p.m.,*
in response to message #1 by Karl Schneider

Karl,

> Quote:
>
> Can anyone provide a direct analytical solution to this problem?

interesting challenge!! I am looking into this approach. So far I have the following:

The initial problem: X+Y+Z=1 is analogous to the sum of the interior angles of a triangle (normalized to PI).
Since A-A-A is not enough to specify a triangle, I declare the triangle to be of unit volume and then solve for the sides using the law of sines, law of cosines, and the volume eqn's of a triangle. (the answers A/DE +B/FG+C/HK are then in radians -> FDISP to get the fractions on my 32sii. More later if time allows. Regards, al

Edited: 15 Apr 2007, 4:22 p.m.

Re: A modest arithmetical challenge

Message #11 Posted by ***Gerson W. Barbosa*** *on 16 Apr 2007, 7:15 p.m.,*
in response to message #1 by Karl Schneider

> Quote:
>
> Can anyone provide a direct analytical solution to this problem?

Dr. Math has provided an analytical solution here, however it doesn't look so direct:

http://mathforum.org/library/drmath/view/56829.html

Gerson.

Edited: 16 Apr 2007, 7:17 p.m.

My C program ("A modest arithmetical challenge")

Message #12 Posted by ***Karl Schneider*** *on 17 Apr 2007, 1:27 a.m.,*
in response to message #1 by Karl Schneider

All --

Thanks to all who have read or taken interest in "A modest arithmetical challenge", and especially to Valentin and Gerson for developing solutions in computer language.

As promised, I'm posting my recursive C-language solution from 12 years ago. I wrote it more as a training exercise than as a fully-optimized product. It performs much redundant processing by generating *every possible* permutation of digits (9! in all) instead of restricting the selections to those that are feasible and mathematically-unique.

The source code would be more compact if it weren't for the comments and two subroutine definitions with required function prototypes and variable declarations. The code is ANSI-standard C (1989), so if you have a C compiler, the copied and pasted code should work just fine.

-- KS

```
/*********************************************
  This program solves the equation


    d1        d4        d7
   ----   +  ----   +  ----   = 1
   d2d3      d5d6      d8d9


where each dn is a UNIQUE digit from 1 to 9.


e.g., 7/12 + 8/36 + 9/45 = 1.005 (181/180)
```

```
      is ALMOST a solution.


This problem appeared in an issue of IEEE
Potentials in early 1994, I believe.


The answer, which you can obtain by compiling
and running this program, is


5/34 + 7/68 + 9/12 = 1.00000


AUTHOR:  Karl Schneider


************************************************/


#include <stdio.h>


void solve (int digit[]);
void next_digit (int i, int digit[], int used[]);


main ()
{
   int digit[10] = {0}, used[10] = {0};


   /* "digit[1] - [9]" holds the 9 assigned digits dn
      "used[1] - [9]" holds flags indicating if
      the corresponding numeral is already in use  */


   next_digit (1, digit, used);


}


/*
"next_digit" is recursive.  Its purpose is to assign
a unique numeral to each of the 9 digits, one digit
at a time, d1 - d9.  There are 9! = 362,880 combinations
of assignments that "next_digit" will generate.
```

```
Each iteration assigns a single digit, and will spawn
another iteration of "next_digit" to assign the next digit
until the 9th digit is assigned.  Then, "solve" will be
called to see if the set of assignments is a solution.
*/


void next_digit (int i, int digit[], int used[])
{
   int d, k;  /* "d" and "k" are numeral counters.
                 "i" is a digit counter.  */


   for (d = 1; d <= 9; d++)
   {
      /*  Clear the "used" array for each numeral used
          by this digit through the 9th digit.  Clear the
          "digit" array for this digit through the 9th
          digit.  (Much of this clearing is redundant, but
          inconsequential.)
      */


      for (k = i; k <= 9; k++)
      {
         used[digit[k]] = 0;
         digit[k] = 0;
      }


      /*  If this numeral is not in use, assign it to
          this digit, mark the numeral as "in use", and
          solve or go to the next digit, as appropriate.
      */


      if ( !(used[d]) )
      {
         digit[i] = d;
         used[d] = 1;
         if (i == 9)
            solve (digit);
         else
            next_digit (i+1, digit, used);
      }
   }
   return;
}


/*
"solve" will specify the three numerators and three
denominators in the equation, and will determine if the
assigned digits provide a solution.  Integer arithmetic,
using common-denominator theory, is used to avoid floating-
point roundoff errors.
*/


void solve (int digit[])
{
   int num1, num2, num3;
   int den1, den2, den3;
   long int numprod;


   num1 = digit[1];
   num2 = digit[4];
   num3 = digit[7];


   den1 = 10*digit[2] + digit[3];
   den2 = 10*digit[5] + digit[6];
   den3 = 10*digit[8] + digit[9];


   numprod = (num1*den2*den3) + (num2*den1*den3) + (num3*den1*den2);


/*  (Debugging code:  will print 362,880 lines if active!)
   printf ("%1i %2i   %1i %2i   %1i %2i  ",num1,den1,num2,den2,num3,den3);
   printf ("numprod = %i \n\n", numprod);
*/


   if (numprod == den1*den2*den3)
      printf ("Solution found!!\n\n%i/%i + %i/%i + %i/%i = 1.00\n\n",
              num1, den1, num2, den2, num3, den3);


      /*  There are six ways to arrange the solution (3 terms, 3!
          arrangements.)  Therefore, 6 "solutions" will be found for
          every legitimate solution.
```

```
     */
   return;
}
```

A physical HP-15C solution in 48 minutes

Message #13 Posted by ***Valentin Albillo*** *on 22 Apr 2007, 2:39 a.m.,*
in response to message #12 by Karl Schneider

Hi, Karl:

I don't want to let people think that the HP-15C isn't powerful enough to solve your puzzle unless it runs some program for several months or weeks, so I took a little time from sleep to post the following admitedly unelegant code which does the proper thing.

The following HP-15C program will find the solution in a deterministic way (i.e., not depending on random factors) and using nothing but several simple heuristics ***in a mere 48 minutes***, when running on a *physical* HP-15C:

```
01 LBL A   27 LBL 3      RCL+ 5      RCL- 7       RCL 5       RCL 1       RCLx 0
    10        RCL 3       /          STO 8        STO I       STO I       RCL+ 5
   STO 0      STO I      STO 9       RCL RAN#     CF  I       CF  I       R/S
    9         F? I        9          RCL+ 9   106 LBL 5   131 LBL 1       RCL 6
   STO 1      GTO 3      STO 6       RCL 6        DSE 5       DSE 1       R/S
   GSB C      SF  I   57 LBL 6       RCL 7        GTO 5       GTO 1       RCL 7
08 LBL 1       9         RCL 6       RCLx 0       RCL 4       RTN         RCLx 0
   RCL 1      STO 4      STO I       RCL+ 8       STO I   135 LBL C   160 RCL+ 8
   STO I   35 LBL 4      F? I         /           CF  I       STO I
   SF  I      RCL 4      GTO 6        +       112 LBL 4   137 LBL 0
    2         STO I      SF  I        1           DSE 4       CF  I
   STO 2      F? I        9          TEST 5       GTO 4       DSE I
14 LBL 2      GTO 4      STO 7       GTO B        RCL 3       GTO 0
   RCL 2      SF  I   65 LBL 7       RCL 7        STO I       RTN
   STO I       9         RCL 7       STO I        CF  I   142 LBL B
   F? I       STO 5      STO I       CF  I    118 LBL 3        9
   GTO 2   43 LBL 5      F? I     94 LBL 7        DSE 3       GSB C
   SF  I      RCL 5      GTO 7       DSE 7        GTO 3       RCL 1
   RCL 1      STO I       44         GTO 7        RCL 2       R/S
   RCL 2      F? I       RCL- 1      RCL 6        STO I       RCL 2
   RCL+ 0     GTO 5      RCL- 2      STO I        CF  I       RCL+ 0
    /         SF  I      RCL- 3      CF  I    124 LBL 2       R/S
   STO RAN#   RCL 3      RCL- 4  100 LBL 6        ISG 2       RCL 3
    9         RCL 4      RCL- 5      DSE 6    126 LBL 2       R/S
   STO 3      RCLx 0     RCL- 6      GTO 6        GTO 2       RCL 4
```

To find the puzzle's solution:

```
FIX 0, GSB A ->  9, R/S -> 12,
         R/S ->  7, R/S -> 68,
         R/S ->  5, R/S -> 34
```

i.e.: **9/12 +7/68 + 5/34 = 1**, which is the one and only solution, found in *just 48 minutes* on my physical, normal speed HP-15C. A 750x emulator would run this program in *less than 4 seconds*.

Some worthwhile programming techniques used:

- Flags are used to keep track of which digits are already in use to avoid repeating them. Notice that when flag 8 gets set, the HP-15C *automatically enters complex mode*, but *this doesn't bother the program in the least* nor does it affect the computations. Matter of fact complex mode is being entered and exited any number of times without the execution being affected. Likewise, when flag 9 is set, the HP-15C enters "blink mode", which is transparent to the user and the program. *All flags are cleared* as soon as the solution is found, so restoring the HP-15C to normal operation upon program's termination.

- In order to find a solution as fast as possible, running speed has been optimized at the expense of code length, so no subroutines are called within the inner loops even if it means duplicating code, and no tricks are played with indirect addressing even if it means replicating whole sections. This gives the fastest possible code.

- Intermediate invariant results are calculated as soon as the variables involved are known and kept in auxiliary registers to avoid recomputing their value in the inner loops. Also, the frequently used value 10 is kept in an auxiliary register, saving program steps and improving execution speed.

- The following simple heuristics are used, as already discussed by myself in a previous post:

  - It's quite easy to prove that in order for the three fractions to add up to 1 or more, it's mandatory that one of the denominators begins by the digit 1 (i.e.: 12, 13, ..., 19). The program makes use of this and, as the order doesn't matter, the first digit of the first denominator is assumed to be 1 without loss of generality.

  - The fact that the corresponding numerator must be 6,7,8, or 9, isn't made use of, but as the first fraction must be large enough for the sum to reach 1, all digits in all fractions are tried in reverse order, from 9 down to 1, except for the first denominator's second digit, which are tried in lowest to highest order. This ensures that the first fraction is as big as possible as early as possible (i.e.: 9/12, 9/13, ..., 9/18, 8/12, 8/13, ..., 8/19, etc).

- Once a solution has been found, the program clears all the flags, outputs the solution, and duly ends.

These same techniques can be applied to create an HP-71B version, which would find the solution in a mere 2 minutes in a physical HP-71B or a few seconds under Emu71. I have absolutely no more time available so that's left as an exercise for the reader. :-)

Best regards from V.

Re: A physical HP-15C solution in 48 minutes

Message #14 Posted by ***Gerson W. Barbosa*** *on 22 Apr 2007, 12:44 p.m.,*
in response to message #13 by Valentin Albillo

Hi Valentin,

That's the HP-15C solution Karl and all of us here were waiting for. That's what we would expect from you, no less!

> Quote:
>
> I don't want to let people think that the HP-15C isn't powerful enough to solve your puzzle unless it runs some program for several months or weeks,

I don't think anyone might have been led to believe this just because of my program. As I said earlier in this thread, a plain brute force program would be able to find the solution within hours, not weeks. I also said I thought I was unable to write one due to its intrinsic complexities. On the other hand, my solution based on random numbers was quite easy to implement, so easy even I was able to do it. My approach would apply only to today's blazingly fast computers though, which can find the solution in less than one second. A quite good compromise between developing and running times, in my opinion.

Best regards,

Gerson.

Re: A physical HP-15C solution in 48 minutes

Message #15 Posted by ***Valentin Albillo*** *on 22 Apr 2007, 6:59 p.m.,*
in response to message #14 by Gerson W. Barbosa

Hi, Gerson:

> Of course, of course, I wasn't trying to belittle your highly imaginative non-deterministic solution, far from it.
>
> Many times Monte Carlo-like methods provide very reasonable solutions for highly intractable problems, or even the only way to solve them, as is the case for multidimensional integration of high dimensionality ("the curse of dimension").

It's only that I got the impression that some people reading this thread could be led to the wrong conclusion that it would take excessively long for an HP-15C to try and solve this puzzle, and I won't suffer such a humiliating situation for my beloved HP-15C if I can do something about it, which I can.

So, a couple of hours later I had written and run the posted program which, indeed, does find the solution in a pretty decent time. A first version using subroutines was very much shorter and elegant, but it took more than twice the time, at some 2 hours. Removing subroutines and unrolling some loops resulted in the 160-step, 48-min program posted, which can be further optimized but to no real purpose. As a 'concept proof' of the HP-15C capabilities, it's already more than adequate.

Thanks for your comments and

Best regards from V.

Re: A physical HP-15C solution in 48 minutes

Message #16 Posted by ***Karl Schneider*** *on 22 Apr 2007, 3:34 p.m.,*
in response to message #13 by Valentin Albillo

Hi, Valentin --

Well, sir, that's quite impressive. I'll definitely run your program on my HP-15C, and perhaps install the emulator program to give it a try there. After I accelerate an HP-15C, I'll quite likely re-visit the program to measure the benefit.

The intelligent heuristics you applied are abolutely necessary to make the program acceptably efficient. My "days/weeks" estimate was admittedly worst-case, as it included exhaustive trial of all possible combinations (or at least, 1/6-th * 9! = 60,480 combinations excluding mathematically-equivalent combinations), assuming no knowledge that only one solution existed, and not simply terminating upon finding one.

An estimated five seconds per trial times 60,480 trials would still make 84 hours, which is unacceptable. Fixing one digit reduces the number of total trials to an eighth thereof, and counting digitsdownward from nine so that larger numerators are tried first is certainly a more intelligent way to find the answer quickly.

Thank you much for your enlightening solution to an HP-15C challenge that has stumped me for years!

Best regards,

-- KS

Edited: 22 Apr 2007, 3:42 p.m.

Re: A physical HP-15C solution in 48 minutes

Message #17 Posted by **Valentin Albillo** *on 22 Apr 2007, 7:27 p.m.,*
in response to message #16 by Karl Schneider

Hi, Karl:

I'm quite glad you like it. It's actually been a very interesting challenge, and I only regret that I really couldn't afford the dedicated time it deserved, which is easily seen in the non-elegant, non-optimal solutions I could accomplish, sorry :-(

By the way, quoting some of your comments in your original post, here is how they apply to my posted solution:

"I believe that the HP-15C offers sufficient programming features to tackle the problem using the brute-force method:

1. *Enough storage registers to maintain nine separate digits and to store solutions"*

   Yes, except that there's no need to store solutions in this case, as there is but one which can simply be output right away.

2. *Nine usable flags (flags 0-9 excluding flag 8) to track whether a digit is "in use"*

   He, he, there's no need either to "exclude" flag 8 as in this case entering and exiting complex mode is transparent to the program's computations. At some times it is actually doing complex-number arithmetic, but the results are of course the same.

   That's one of the most beautiful advanced characteristics of the HP-15C, the natural, almost transparent way in which it handles complex-number calculations, which very frequently require *the very same keystrokes/program-steps* as their real-valued counterparts.

3. *Subroutines, conditional tests, and plenty of labels*

   No subroutines are called within the main loops, for speed, and nearly all numeric labels are reused *two times* (except LBL 2, which appears *three* times, one of them as a mere placeholder for the ISG instruction). This extremely convenient reutilization of labels would be *impossible* in an HP33S, say.

4. *A versatile indirect register*

   Yes, but having the equivalent of SF IND X, CF IND X, FS? IND X, as in the HP-41C, would have resulted in a much shorter, clearer, and faster program. Alas, flag indirection can only be done via the I register in the HP-15C, not the X register.

So, a workable HP-15C program could be written. However, I also believe that such a program might take literally weeks to run to completion on an actual HP-15C!"

In the end, with a few sensible (and easy-to-justify) heuristics here and there, it actually could be done in less than an hour. I expect this will increase even further your already high appreciation of this most awesome model.

Best regards from V.

A physical HP-71B solution under 20 minutes

Message #18 Posted by ***Gerson W. Barbosa*** *on 24 Apr 2007, 12:28 a.m.,*
in response to message #13 by Valentin Albillo

Hello again Valentin,

> Quote:
>
> These same techniques can be applied to create an HP-71B version, which would find the solution in a mere 2 minutes in a physical HP-71B or a few seconds under Emu71. I have absolutely no more time available so that's left as an exercise for the reader. :-)

I didn't try your technique but I did try another solution on the HP-71B.

Accepting, without proving, the relationship provided by Karl, that is,

```
D=2*B
```

it's easy to obtain

```
          D*E
F = - --------------
      2*( A - B) + C
```

The program also assumes A is even, as Karl has suggested.

The program runs in about 11 second on Emu71 (@500MHz), however the unique answer is shown after 6 seconds. On the real HP-71B the total running time was exactly 19 minutes and 58 seconds (Well, that's under 20 minutes :-) As always, my best solutions are 10 to 20 times slower than yours, but I am glad I was able to come up with something that would not take days or weeks on the real calculator :-)

Best regards,

Gerson.

```
------------------------------------------------------------------
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(8),M(8) @ FOR I=0 TO 8 @ M(I)=I+1 @ NEXT I
15 FOR A=1 TO 9 STEP 2 @ FOR C=1 TO 9 @ IF C=A THEN 75
20 FOR E=1 TO 9 @ IF E=C OR E=A THEN 70
25 FOR B=12 TO 49 @ IF MOD(B,5)=0 OR B=A+C/2 THEN 65
30 D=2*B @ F=-D*E/(2*(A-B)+C) @ IF FP(F)<>0 OR F>99 OR F<12 OR F=B THEN 65
35 IF MOD(F,10)<>0 THEN GOSUB 80
40 K=1 @ I=0
45 S=0 @ I=I+1 @ FOR J=0 TO 8 @ IF M(I)=N(J) THEN S=S+1
50 NEXT J @ IF S<>1 THEN K=0
55 IF I<8 AND K THEN 45
60 IF K THEN PRINT A;"/";B;"+";C;"/";D;"+";E;"/";F;" =  1"
65 NEXT B
70 NEXT E
75 NEXT C @ NEXT A @ END
80 N(0)=A @ N(1)=INT(B/10) @ N(2)=MOD(B,10) @ N(3)=C @ N(4)=INT(D/10)
85 N(5)=MOD(D,10) @ N(6)=E @ N(7)=INT(F/10) @ N(8)=MOD(F,10) @ RETURN
------------------------------------------------------------------


>RUN
 5 / 34 + 7 / 68 + 9 / 12  =  1
```

Here is the equivalent Turbo Pascal program, for clarity:

```
--------------------------
Program Karl_Problem;
var a, b, c, d, e, f, i, j: byte;
                   x, r, s: real;
                        ok: boolean;
                      m, n: array [1..9] of byte;
begin
  ClrScr;
  for i:=1 to 9 do
    m[i]:=i;
  a:=-1;
  repeat
    a:=a+2;
    for c:=1 to 9 do
      if c<>a then
        for e:=1 to 9 do
          if (e<>c) and (e<>a) then
          for b:=12 to 49 do
            if ((b mod 5)<>0) and (b<>(a+c/2)) then
              begin
```

```
                  d:=2*b;
                  x:=-d*e/(2*(a-b)+c);
                  if (Abs(Frac(x))<0.000001) and (x<99) and (x>11) and (x<>b) then
                    begin
                      f:=Trunc(x);
                      if f mod 10 <> 0 then
                        begin
                          n[1]:=a; n[2]:=b div 10; n[3]:=b mod 10;
                          n[4]:=c; n[5]:=d div 10; n[6]:=d mod 10;
                          n[7]:=e; n[8]:=f div 10; n[9]:=f mod 10
                        end;
                      ok:=true; i:=0;
                      repeat
                        s:=0; i:=i+1;
                        for j:=1 to 9 do
                          if m[i]=n[j] then
                            s:=s+1;
                        if s<>1 then
                          ok:=false
                      until (i=9) or (not ok);
                      if ok then
                        WriteLn(a:1,'/',b:2,'+',c:1,'/',d:2,'+',e:1,'/',f:2,'=1')
                    end
                end
  until a=9
end.


--------------------------
5/34+7/68+9/12=1
--------------------------
```

Edited: 24 Apr 2007, 2:14 p.m.

HP-15C program

Message #19 Posted by ***Gerson W. Barbosa*** *on 17 Apr 2007, 8:45 p.m.,*
in response to message #1 by Karl Schneider

Quote:

I believe that the HP-15C offers sufficient programming features to tackle the problem using the brute-force method:

Enough storage registers to maintain nine separate digits and to store solutions Nine usable flags (flags 0-9 excluding flag 8) to track whether a digit is "in use" Subroutines, conditional tests, and plenty of labels A versatile indirect register So, a workable HP-15C program could be written. However, I also believe that such a program might take literally weeks to run to completion on an actual HP-15C! Recursion and integer arithmetic could not be employed, as my C program does.

The HP-15C program below might find the solution, given enough time (five days? one week? one fortnight?). Of course, this depends also on the quality of the random number generator.

As a test, the line 57 could be replaced with .25 and the line 58 with g TEST 8. If the program has been keyed in correctly, the answer will be 439,265,871.0, that is, 4/39 + 2/65 + 8/71. This is equal to 0.2460093897, the first sum less then 1/4 found by the program. This takes less than 60 seconds on my 2.2x 15C.

Line numbers followed by a lower case "u" have to be keyed in USER mode.

This has been written just an exercise to remember the use of matrices on the HP-15C. It is far from optimized. Improvement suggestions are welcome. Besides pure brute force, the program relies on luck, which is not good, unless, of course, we are lucky enough :-)

The program is just a translation of the previous QBASIC program to RPN. The second matrix is not initialized to zero because this is not necessary.

Regards,

Gerson.

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


001-    f LBL B
002-    1
003-    ENTER
004-    9
005-    STO RAN #
006-    f DIM A
007-    f DIM B
008-    f MATRIX 1
009-    f LBL 2
010-    RCL 1
011u    STO A
012-    GTO 2
013-    1
014-    STO 2
015-    f LBL 7
016-    f RAN #
017-    9
018-    *
019-    1
020-    +
021-    g INT
022-    STO 1
023-    RCL A
024-    g x=0
025-    GTO 5
026-    0
027-    STO A
028-    Rv
029-    RCL 2
030-    STO 1
031-    x<>y
032-    STO B
033-    1
034-    STO+ 2
035-    f LBL 5
036-    RCL 2
037-    9
038-    g TEST 9
039-    GTO 7
040-    f CLR SIGMA
041-    f MATRIX 1
042-    RCL 1
043-    f LBL 8
044-    +
045-    /
046-    +
047u    RCL B
048u    RCL B
049-    1
050-    0
051-    *
052u    RCL B
053-    GTO 8
054-    +
055-    /
056-    +
057-    1
058-    g TEST 6
059-    GTO 2
060-    0
061-    f LBL 9
062-    *
063-    STO+ 2
064-    RCL 1
065-    CHS
066-    9
067-    +
068-    10^x
069u    RCL B
070-    GTO 9
071-    *
072-    STO+ 2
073-    RCL 2
074-    g RTN
```

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


Estimated runtime:


((9! * 40s * 1.5)/86400s)/2 = 126 days! (or up to 252 days)


(Assuming each combination takes 40 seconds to process, and there are 50% of repeated combinations).
 I would recommend a voltage adapter since the batteries would last only 60 hours, according
 to the manual :-)
```

Edited: 17 Apr 2007, 10:58 p.m.

Re: HP-15C program

Message #20 Posted by ***Karl Schneider*** *on 19 Apr 2007, 12:20 a.m.,*
in response to message #19 by Gerson W. Barbosa

Hi, Gerson --

Thank you for the imaginitive HP-15C solution. I'll have to give it a try on an accelerated calculator or emulator.

I have to admit that an approach utilizing random numbers never would have occurred to me...

Best regards,

-- KS

Re: HP-15C program

Message #21 Posted by ***Gerson W. Barbosa*** *on 19 Apr 2007, 12:43 p.m.,*
in response to message #20 by Karl Schneider

Hello Karl,

> Quote:
>
> ---
>
> I'll have to give it a try on an accelerated calculator or emulator.

I fear this might not be a good idea. The average number of tries before the solution was found was 91,985.8 in 50 runs of a modified QBASIC program. In the first run the solution was found after 166,192 tries while in the second run the answer appeared after only 3,394 tries (this was the least number of tries; the maximum number of tries was 256,069, in the 23rd run). Assuming my sped-up 15C takes 20 seconds to process each try, this would take from 19 hours to 60 days to run (from 42 hours to 131 days on a normal 15C!).

I wish there were a fast running mode in Nonpareil :-)

Regards,

Gerson.

P.S.: This modification saves one step and might cause the solution to be found weeks earlier or later :-)

```
001-    f LBL B
002-    1
003-    STO RAN #
004-    9
005-    f DIM A
        .
        .
        .
```

The estimated runtime I provided earlier was wrong. The correct average runtime should be:

(9!/3!)*40s/86400 = 28 days

(Assuming it takes 40 seconds in average to generate each set of 9 different digits, which I haven't checked yet).

A feasible way of solving the problem on the HP-15C would be running the program simultaneously on, say, 15 calculators (or Nonpareil instances), each beginning with a different seed. It is probable at least one of them would complete the task in less than 30 hours...

A program using plain brute force would be longer, I guess, and would take about the same time to reach the solution. However, I think I wouldn't be able to write one...

Anyway, considering an HP-15C was recently given the task to compute pi using a Monte Carlo method, the idea of using random numbers to find those three fractions don't look so absurd :-)

Edited: 21 Apr 2007, 8:25 a.m.

Re: HP-15C program

Message #22 Posted by ***David Jedelsky*** *on 20 Apr 2007, 6:15 a.m.,*
in response to message #19 by Gerson W. Barbosa

Hi, it really works. I tried it in my testing simulator of 15c and got answer in about 68min of cpu time. If my estimate is correct it roughly corresponds to 850hours on real calculator. Here is the result:

```
      --   --        --        --   --   --   --   --
     |       | |  | |  |    |    |    | |    |  | |  |
      --   --   --   --        --        --   --
        |    |    |    |    | |       | |  | |  | |  |
      --   --      , --        -- ,      --   -- . --


real    75m56.331s
user    67m49.726s
sys     0m22.688s
```

Best Regards,

David

Re: HP-15C program

Message #23 Posted by ***Gerson W. Barbosa*** *on 20 Apr 2007, 12:34 p.m.,*
in response to message #22 by David Jedelsky

Hi David,

Thanks for taking the time to test the program on your simulator. I have an instance of Nonpareil running since yestarday (seed=1); my sped-up 15C (2.2x) ran for at least 52 hours before the batteries went down (not fresh alkaline batteries). By the way, is your simulator freely available? Does it run under DOS or Linux? Is it based on Eric Smith's Nonpareil? 750x is quite good :-)

The modification below, beginning from line 40 would save about two hours in a month, on a real HP-15C. Optimizing the first part of the program would give a better result though.

```
039-    GTO 7           058-u   RCL B
040-    f MATRIX 1      059-u   RCL B
041-u   RCL B           060-    1
```

```
042-u   RCL B           061-    0
043-    1               062-    *
044-    0               063-    RCL B
045-    *               064-    f MATRIX 1
046-u   RCL B           065-    +
047-    +               066-    /
048-    /               067-    +
049u    RCL B           068-    1
050u    RCL B           069-    g TEST 6
051-    1               070-    GTO 2
052-    0               071-    0
053-    *               072-    STO 2
054u    RCL B           071-    f LBL 9
055-    +                .
056-    /                .
057-    +                .
```

One could use the simulator to try different seeds (twenty of them would suffice) until the solution was found under 20 hours on the real calculator. Of course, this would be cheating :-)

Best regards,

Gerson.

Re: HP-15C program

Message #24 Posted by ***David Jedelsky*** *on 20 Apr 2007, 5:33 p.m.,*
in response to message #23 by Gerson W. Barbosa

Hi Gerson,

I appreciate your interest in my simulator, but I'm afraid it is not much of general usability. I created it just as a reference platform for the testing of my PIC simulator core. It runs under linux and I even used one source file (digit_ops.c) from Nonpareil (thank you Eric), because the implementation of arithmetic operations was clear and it saved me some time. It is pure console application, as you can also see from the printed result in my previous post. Moreover, it accepts keys just by internal calculator codes. So, it is not mentioned as simulator for the general calculator use. It is just testing tool. Anyway, if anybody want to use it or take a look at it I'm prepared to publish it.

Regarding the speed. I don't think 750x is really good, but definitely enough :). If I include optimization options to gcc and place my cpu at regular speed 2.2GHz it runs even somewhere around 1000x. But the code is far from optimal. I think it is possible to reach several times better speeds with optimizations.

I didn't try your modification so far. But at least tried seed bases from 0.00 to 0.99 ;). The winner is 0.12 (12 STO RAN#). My estimation for your 2.2x calculator is around two hours to get the result (but maybe I'm wrong).

Well, you are right, it is kind of cheating, but who don't want to see real result after all this effort :-).

Best Regards,

David

Re: HP-15C program

Message #25 Posted by ***Gerson W. Barbosa*** *on 21 Apr 2007, 12:21 a.m.,*
in response to message #24 by David Jedelsky

Hi David,

Thanks again for testing.

> Quote:
>
> My estimation for your 2.2x calculator is around two hours to get the result (but maybe I'm wrong).

You are right! I forgot to check the chronometer: last time I had checked about 1 hour and 50 minutes had elapsed. When I remembered to check it again 15 minutes later the calculator was still on and displaying 912.768.534,0. So, we can conclude it will take about about 4 hours and 30 minutes on a normal 15C if the following modification is made in the original program:

```
001-    f LBL B
002-    1
003-    2
004-    STO RAN #
005-    g LOG
006-    9
007-    f DIM A
        ...
```

This 75-step program may not be fastest one to solve Karl's problem on the 15C, even considering the cheating, but surely is one of the shortest ;-)

Just a curiosity, from 1000 runs of the equivalent Turbo Pascal program:

```
...
 7 6 8 9 1 2 5 3 4  993    112691
 9 1 2 7 6 8 5 3 4  994     30159
 9 1 2 5 3 4 7 6 8  995      5370
 9 1 2 5 3 4 7 6 8  996    175788
 5 3 4 9 1 2 7 6 8  997      6076
 9 1 2 7 6 8 5 3 4  998      1388
 7 6 8 9 1 2 5 3 4  999     59464
 9 1 2 7 6 8 5 3 4 1000      7443


mean:   61158.5180   min:  8  max: 422667
```

It is possible there is a magic seed that allows the solution to be found in only eight tries on the 15C as well, that is, in about five minutes! On the other hand, 422667 tries would require at least 6 months...

Best regards,

Gerson.

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

P. S.:

This run is even more amazing:

```
...
 9 1 2 7 6 8 5 3 4  998     88848
 9 1 2 7 6 8 5 3 4  999      3748
 5 3 4 7 6 8 9 1 2 1000     40377


mean:   61250.5890   min:  1  max: 593567
```

At least once the solution was found in the first try...

Edited: 21 Apr 2007, 12:42 a.m.

[Return to Index | Top of Index]