♥MoHPC♠ The Museum of HP Calculators

HP Forum Archive 14

[Return to Index | Top of Index]

A New Small HP-15C Challenge (Prize awarded!) [LONG]

*Message #1 Posted by **Valentin Albillo** on 15 June 2004, 7:37 a.m.*

Hi all,

To spice the week a little, exclusively for HP-15C owners and lovers (i.e., most of us; no other calculators allowed), here's a **New, Small HP-15C Challenge**. As always, it's very small yet it tries hard to be both entertaining and educational as well. But this time there are a couple novelties:

- You must have *a real pro attitude* this time. You'll be "commissioned" to develop a solution *strictly* according to a series of compulsory requirements, and you must deliver, as in real life.
- There's a (small) Prize awarded this time to the person who writes the best routine which unfailingly meets all specifications. 'Best' is intended to mean the routine is as short, fast, and uses the least resources as possible. If several solutions are equally deserving, all with be awarded the prize, myself being the final and sole judge.

The (small) Prize is your choice of a complete, very high-quality scan of your preferred *rare* vintage HP brochure to choose among these three:

```
HP-65 brochure (in English, rare!)
HP-55 brochure (in Spanish, very rare!!)
HP-01 brochure (in French, extremely rare!!!)
```

You can click the above links to see (purposely low-quality, lo-res) samples of all three. The scanned brochures themselves are marvelous, large, hi-res scans, multiple pages for each brochure, to be delivered to you electronically. They're certainly a pleasure to behold and read (though it also makes you a little sad and nostalgic to see what superb products and marketing brochures HP produced in its prime ...)

Even if you're not (!?) that interested in the scanned brochures, have a go at the Challenge. Though it seems simple enough, you'll quickly realize it's far from trivial to come up with even a working version according to the specifications, let alone an optimum one.

The Challenge

The owner of a small business wants, given a certain discount percentage rate between 10.00% and 99.99% (both included), to compute both discounted amounts from a series of non-discounted amounts and vice versa. She also wants to be able to total both types of amounts for a series of conversions, plus be able to specify the percentage rate at any time.

To that effect she's commissioned you to write a routine (LBL A ... RTN) for her trusty HP-15C, which will be called either from the keyboard or from some other main program, and must *EXACTLY* meet the following specifications, to-the-letter.

1. **To specify the percentage rate** (P), your routine will be called with the percentage (between 10.00% and 99.99%) in the X register, will likely store it somewhere else for later computations, and must return with that same value unchanged in the X register, like this:

   To set a 10.00% discount:

   ```
   10 [A]  ->  10.0000
   ```

2. **To convert non-discounted amounts to discounted amounts**, your routine will be called with matrix C (which will hold a certain number of non-discounted amounts, arbitrary dimensions) in the X register. . Your routine must replace them in-place with the discounted amounts and return with said matrix C in the X register, like this:

   Assuming that P = 10.00% and with C = (20, 30, 50), to compute the discounted amounts:

   ```
   RCL MATRIX C [A]  ->  [C  1 3]
   ```

   where now C holds the discounted amounts, C = (18, 27, 45)

3. **To convert discounted amounts to non-discounted amounts**, your routine will be called with matrix D (which will hold a certain number of discounted amounts, arbitrary dimensions) in the X register. Your routine must replace them in-place with the non-discounted amounts and return with said matrix D in the X register, like this:

   Assuming that P = 10.00% and with D = (36, 63, 81), to compute the original, non-discounted amounts:

   ```
   RCL MATRIX D [A]  ->  [d  1  3]
   ```

   where now D holds the original, non-discounted amounts, D= (40, 70, 90)

4. **To totalize the discounted amounts in matrix C to the grand totals stored in matrix E**, your routine must expect matrix C to be in Y and matrix E to be in X, like this:

   ```
   RCL MATRIX C, RCL MATRIX E [A]  ->  [E  1 3]
   ```

   Assuming C = (18, 27, 45) and that E was (10, 20, 30), E is now (28, 47, 75)

5. **To totalize the original, non-discounted amounts in matrix D to the grand totals in matrix E**, your routine must expect matrix D to be in Y and matrix E to be in X, like this:

```
         RCL MATRIX D, RCL MATRIX E [A]  ->  [E  1 3]
```

   Assuming D = (40, 70, 90) and that E was (15, 20, 25), E is now (55, 90, 115)

Notes:

- in (4) and (5) above, notice that clearing matrix E before totalling is *not* the responsibility of your routine. This will be done in the calling program or from the keyboard [0, STO MATRIX E] if desired, it's not a functionality that your routine must implement.
- you must not assume that the stack (X,Y,Z,T,Last X) will hold any specific contents before calling your routine other than the exact required parameters specified above. Same goes for storage registers' contents, flags setting or modes (except for the place where you stored the percentage rate, which is assumed to still be there, of course).
- you must not assume any specific storage register allocation [DIM (i)], so your routine must work on the assumption that only the permanently storage registers are allocated.
- and last, but certainly not least, it should be clear by now that LBL A is the *one and only* entry point to your routine. The calling program will always use **GSB A** to invoke all your routine's functionalities or else **[A]** will be used from the keyboard. Your routine must then assume that it will always be entered at LBL A, with the required parameters on the stack as per the desired function's specifications.

  It shall also assume that all variables and matrices are named *exactly* as specified and contain the specific kinds of amounts or totals described above.

Well, let's see if you're up to the task. Of course, the least resources your routine uses, the better. There's a solution fulfilling all specified requirements in 24 program steps or less (including LBL A ... RTN), which I will post after the Challenge is over, within a few days. Hope you'll enjoy it and ...

Best regards from V.

Re: A New Small HP-15C Challenge (Prize awarded!) [LONG]

Message #2 Posted by ***Bram*** *on 16 June 2004, 3:44 a.m.,*
in response to message #1 by Valentin Albillo

> Quote:
>
> and must *EXACTLY* meet the following specifications, to-the-letter

You won't allow me to cheat again, will you? ;-)
Well, don't worry; I don't have an HP-15C (still waiting for Chris).
However, I did think about the assignment and I therefor did read the manual from Dave's CD set, but I just need the feed back from the real device to test and evaluate the solution(s).

(waiting for the next (non-matrix) problem -)

Re: A New Small HP-15C Challenge (Prize awarded!) [LONG]

Message #3 Posted by ***RMillán*** *on 16 June 2004, 11:57 a.m.,*
in response to message #1 by Valentin Albillo

Well, here is my try:

```
 01 ·f LBL A
 02  f MATRIX 7
 03  GTO B
 04  STO RAN#
 05  RTN
 06 ·f LBL B
 07  CLx
 08  LSTx
 09  STO I
 10  STO RESULT
 11  1
 12  RCL RAN#
 13  -
 14  GTO I
 15 ·f LBL D
 16  1/x
 17 ·f LBL C
 18  *
 19  RTN
 20 ·f LBL E
 21  Rdown
 22  +
 23  RTN
```

Notes:

1. Register RAN# can only be used, and saves a few lines, because of the discount percentage is >=10% and <100%. Storing it in RAN#, which zeroes the exponent for numbers >=1, effectively divides it by 100. This trick wouldn't work for numbers smaller than 10.

2. I don't know if indirect label addressing with matrix indicators is documented. I tried it and it worked.

3. The only thing I don't like about this solution is leaving the number 1 - P/100 in registers Z and T for cases 4 and 5, but, as long as I understand the challenge requirements, it is allowed.

4. I did it in 23 steps, so I'm afraid I made some mistake. But it seems to work as required...

5. Wow! I hadn't programmed with keycodes for a long time!

Best regards from
Rafael Millán.

Edited: 16 June 2004, 12:08 p.m.

Re: A New Small HP-15C Challenge (Prize awarded!) [LONG]

Message #4 Posted by ***Valentin Albillo*** *on 21 June 2004, 11:47 a.m.,*
in response to message #3 by RMillán

Hi, Rafael:

Rafael posted:

> **A perfectly correct solution !**

to my little HP-15C challenge. Congratulations, your solution is virtually identical to mine, with the improvement of being one step shorter by the neat trick of making a routine fall down on another and thus saving a RTN instruction !!.

Just for the record, here is my original solution, commented:

```
01  LBL A       the mandatory entry point
02  MATRIX 7    we need to discriminate if we're being passed a matrix
                or an scalar. The HP-15C has two instructions which can
                do that, namely MATRIX 7 (the Row Norm) and MATRIX 8 (the
                Frobenius Norm). Both take a matrix in X and return the required
                norm, and both skip the next step in program memory if the
                value in X is not a valid matrix descriptor. I use
                MATRIX 7 because evaluating the Row Norm is faster.
03  GTO 0       The value in X was a matrix, branch to deal with it.
04  STO RAN#    The value in X was an scalar; as it should be in the range
                10.00 to 99.99, we can store it in the RAN# register, which
                saves a register and further normalizes the number to the
```

```
                    range 0.1000-0.9999, thus effectively dividing it by 100.
    05  RTN         percentage stored; return to the calling program
    06  LBL 0       internal entry point for matrix processing
    07  RDN         get rid of the unneeded Norm
    08  LAST X      bring back the given matrix, which was stored in LAST X
                    when MATRIX 7 computed the Row Norm.
    09  STO I       store the matrix descriptor in Register I, for later indirect
                    addressing
    10  STO RESULT  specify this same matrix to hold the result
    11    1         to compute 1-Percentage ...
    12  RCL RAN#    ... recall normalized percentage
    13    -         done with the computation
    14  GTO I       with a matrix descriptor in Register I, this line
                    transfers execution to the label indirectly specified
                    by the same letter as the matrix descriptor.
    15  LBL C       matrix C was specified in X
    16   *          MAT C = C * (1-Percentage)
    17  RTN         done; return to the calling program
    18  LBL D       matrix D was specified in X
    19   /          MAT C = C / (1 - Percentage)
    20  RTN         done; return to the calling program
    21  LBL E       matrix E was specified in X
    22  RDN         get rid of the percentage
    23   +          MAT E = E + C or E + D
    24  RTN         done; return to the calling program
```

Rafael's solution saves one step (RTN) and another RTN can be saved at the very end, so the specifications can be met with just a 22-step routine. I think this is a very fine example of just how powerful and comprehensive the HP-15C's instruction set is, not to mention, if I may coin a word, *"finetuneable"*. :-)

I've enjoyed creating this challenge, and sincerely hope you enjoyed solving it or reading the solution's details. I would certainly have preferred to gather more inputs, but I guess Rafael's solution, being so close to my own, it's proof enough that the solution is *unique* and so, after Rafael posted his, noone could improve on it or give an alternate one. Congratulations, again !

As for you, Rafael, please tell me what's your choice brochure and a valid e-mail address for me to send it, or else you can download it from a site of mine which I'll tell you privately if you would provide an usable address.

Best regards from V.

Re: A New Small HP-15C Challenge (Prize awarded!) [LONG]

Message #5 Posted by ***RMillán*** *on 22 June 2004, 1:53 p.m.,*
in response to message #4 by Valentin Albillo

Valentín:

I must say that the challenge requirements were so narrow that I was literally pushed towards the solution:

- Just an entry point? So, there must be a function which can tell a matrix from a number. I looked for and I found it. I didn't know before there was one.
- So few program lines suggested some kind of indirect addressing, but it would have to work with matrix descriptors, which I would have never tried otherwise. I tried, and it worked. Also, I didn't know this before.
- Etc.

Of course, I enjoyed the challenge: I learned so much in just one hour.

I think I will go for the HP65 brochure. Please, send me the address for downloading the files to ramb arroba textodigital punto com.

Best regards,
Rafael Millán.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall