05-04-2018, 11:39 PM **Post: #1**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**[VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" Special**

.

**Welcome to my *"Star Wars"*-themed *S&SMC#23 "May the 4th Be With You" Special***, young *padawan*.

Here a difficult challe.. erm, *"Jedi Trial"* awaits you. Should you complete its ***6 Steps*** to the complete satisfaction of your *Jedi Master* (that would be me..), you'll be a lowly *padawan* no more but you'll be promoted to be the ***71st Jedi Knight***.

Alas, it won't be easy ! Far from it ! A great reward requires great achievements. But should you succeed, you'll have learned some ancient, valuable techniques which will vastly enrich your skills and, dare I say it ? Yes ! You'll even have Great Fun !

Shall we begin ? But first of all, a little couple of

**Notes**: While the final *6th Step* is solvable to some degree or other with most any reasonably advanced HP calcs, the first *5 Steps* are intended specifically for the **HP-71B**. It might be the case that some of them aren't meaningless and might be solved using other HP models but I can't be sure.

When using an **HP-71B**, unless the particular task specifies otherwise (and most do), you can also use any of these ROMs: *Math, HPIL, JPC,* plus the *STRNGLEX* Lex file. No other ROMs or LEX files allowed.

In any case, you must use either a *physical* or *emulated/simulated* **HP calc**, solutions for other devices aren't allowed.

## Step the First:

Write a program which accepts from the user an integer N from 1 to 19 and outputs the Nth digit of *Log(10)*, the natural logarithm of 10 (= *2.302585092994045684* to 19-digit accuracy).

*Example*:

- if the user specifies **1**, it must output **2** (the *1st* digit of Log(10))
- if the user specifies **2**. it must output **3** (the *2nd* digit of Log(10))
- if the user specifies **18**, it must output **8** (the *18th* digit of Log(10))
- if the user specifies **19**, it must output **4** (the *19th* digit of Log(10))

*Requirements*:

- the shorter and faster, the better
- it must run in a *barebones* HP-71B (no ROM/LEX files allowed)
- it must use *no variables at all*
- and needless to say, you can't supply the full 19-digit value of Log(10) to the program in any way or shape (e.g.: as a string in the program, DATA statements, reading from a file, input from the user, etc.).

I'll post my original solution, which is a 1-line program (48 bytes).

## Step the Second:

If you succeeded with the previous *Step*, you'll find this one dead easy, namely:

Write a program which accepts from the user an integer N from 1 to 32 and outputs the Nth digit of *Pi/2* (= *1.57079632679489661923132169163975* to 32-digit accuracy).

*Example*:

- if the user specifies **1**, it must output **1** (the *1st* digit of Pi/2)
- if the user specifies **2**, it must output **5** (the *2nd* digit of Pi/2)
- if the user specifies **31**, it must output **9** (the *31th* digit of Pi/2)
- if the user specifies **32**, it must output **7** (the *32th* digit of Pi/2)

*Requirements*:

- the shorter and faster, the better
- it must run in a *barebones* HP-71B (no ROM/LEX files allowed)
- it must use *no standard math functions* and *no arithmetic operations* except **+** or **-**
- and again, you can't supply the full 32-digit value of Pi/2 to the program in any way or shape (e.g.: as a string in the program, DATA statements, reading from a file, input from the user, etc.).

I'll post my original solution, which is a 3-line program (152 bytes).

## Step the Third:

Now for something different: you don't have to write a program but instead solve right from the command line the following equation (which actually is a polynomial equation in disguise):

$$\sqrt{x+1} + \sqrt{x+2} + \sqrt{x+3} + \ldots + \sqrt{x+98} + \sqrt{x+99} + \sqrt{x+100} = 700$$

*Requirements*:

- the faster and shorter (in that order), the better
- you can execute more than one command line in succession if need be
- you *can't use data files*
- you *can't run, call or use* any program code whatsoever
- for timing-comparison purposes, use **0** as any initial guess(es)

I'll post my original solution, which is 117 characters long (about 80 bytes). It's slightly longer (just 6 extra characters) but *much* faster (2.2x) than another shorter version which I'll also post.

## Step the Fourth:

After that much trouble to complete the previous *Steps*, now for an easy one:

Write a program which accepts a positive integer N from the user and outputs both the number and its square for every value from N down to 0, both included, one pair per line.

*Example*: assuming the user supplied the number **1234**, the output would be like this, no more, no less:

```
1234    1522756
1233    1520289
1232    1517824
```

```
1231      1515361
    ...
4      16
3      9
2      4
1      1
0      0
```

subject to these

*Requirements*:

- the shorter the better
- it must run in a *barebones* HP-71B (no ROM/LEX files allowed)
- it must use *no variables at all* and no PEEK/POKE either

I'll post my original solution, which is either a 1-line, 49-byte program or a 2-line, 46-byte one.

## Step the Fifth:

Enough with the easy stuff. Now we're getting tougher so you must ...

Write a program which accepts from the user a *single-digit* **Id**, then accepts from the user a **text** to scan for said *Id* and output the name associated with that *Id*.

The format of the text to scan (up to 80 characters long, say) is as follows:

*(Id1):(Name1),(Id2):(Name2), ... , (IdN):(NameN)*

where *(Id)* is a single digit and *(Name) i*s a string of up to 30 characters A-Z & spaces. The *Id* aren't necessarily in numerical order in the *text*, but the *Id* sought for must appear somewhere within the *text*.

*Example*: suppose the user supplies to the program these *Id* and these *texts* to scan (all identical for this particular example):

*Id* = **1**,
*Text* = "*2:Yoda,1:Luke Skywalker,5:Obi Wan Kenobi,3:Darth Vader,4:R2D2*"

Output: ***Luke Skywalker***

*Id* = **5**, same *text* as before Output: ***Obi Wan Kenobi***
*Id* = **2**, same *text* as before Output: ***Yoda***
*Id* = **4**, same *text* as before Output: ***R2D2***
*Id* = **3**, same *text* as before Output: ***Darth Vader***

*Requirements*:

- the shorter the better
- it must run in a *barebones* HP-71B (no ROM/LEX files allowed)
- it must use *no variables at all* and no PEEK/POKE either

Sounds familiar, uh ? I'll post my original solution, which is a 2-line program (74 bytes).

## Step the Sixth:

We'll call a *"Selfie"* to any positive N-digit integer number which has the property that if you sum its N digits raised to the Nth power you get the original number *backwards*. For instance, the **7**-digit number **5271471** is a *Selfie*:

**5271471** => $5^7 + 2^7 + 7^7 + 1^7 + 4^7 + 7^7 + 1^7$ = **1741725**, which is 5271471 *backwards*

Write a program to find all *Selfies* from 1 to 9 digits long (for 10-digit HP calcs, 29 in all) or from 1 to 11 digits long (for 12-digit HP calcs, 37 in all). 0 is *not* a positive number so it's not a *Selfie*.

*Requirements*:

- the faster and shorter (in that order), the better

I'll post my original solution for the **HP-71B**, an 11-line program (398 bytes) which, when run in **Emu71**, finds in 3'20" all 37 *Selfies* up to 11 digits long. I'll also post a 12-line version which is 20% faster, in particular it finds all 11-digit *Selfies* in just over 80 seconds.

That's it, young *padawan*, your Ordeal has come to an end. I'll post my original solutions next Thursday so you've got plenty of time to develop your own. If you succeed in completing the ***6 Steps*** you will become ***the Most Honored 71st Jedi Knight***.

**May the 4th Be With You !!**

Regards.
V.
.

PM FIND EDIT QUOTE REPORT

05-05-2018, 12:00 PM **Post: #2**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

Let's start with the easy part:

> **Valentin Albillo Wrote:** ➡ (05-04-2018 11:39 PM)
>
> Step the Fourth
> ...
> Write a program which accepts a positive integer N from the user and outputs both the number and its square for every value from N down to 0, both included, one pair per line.
> ...
> it must use no variables at all and no PEEK/POKE either

The challenge here is to use no variables (otherwise it's trivial).

Re-using a previous idea, here is my **41**-byte solution:

```
10 DISP VAL(DISP$);RES*RES
20 DISP SQRT(RES)-1;RES*RES @ IF RES THEN 20
```

To use it, type the number N, DON'T press ENTER but RUN.

J-F

EMAIL PM WWW FIND QUOTE REPORT

05-05-2018, 05:44 PM **Post: #3**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

**Valentin Albillo Wrote:** (05-04-2018 11:39 PM)

Step the First:
Write a program which accepts from the user an integer N from 1 to 19 and outputs the Nth digit of *Log(10)*, the natural logarithm of 10 (= *2.302585092994045684* to 19-digit accuracy).
...
Step the Second:
...
Write a program which accepts from the user an integer N from 1 to 32 and outputs the Nth digit of *Pi/2* (= *1.57079632679489661923132169163975* to 32-digit accuracy).

I don't have solutions right now for the Steps the First and Second, but the solutions will be in some way related to the LOG(10) and PI values internally known by the HP71.

PI/4 is known with 31 digits, see for instance here.
and it's easy to make the digits 13-24 of PI visible with SIN(3.14159265358).

LOG(10) is known with 20 digits (2.30...56840) within the EXP function code:

```
Saturn Assembler     Math Routines - Part 1 <831213   Fri Dec 30, 1983   3:18 am
Ver. 3.39/Rev. 2306                                                    Page  66



3535            **********************
3536            *   |X| >= 1  case   *   Double Reduction by ln10 (x'=x-n'*ln10)
3537            **********************
3538
3539 0CFC0 120   DXP200 AROEX                R0=0...XPON(X); A=X_low
3540 0CFC3 AFC          ABEX   W             A=X_high; B=X_low
3541
3542            *   Extra Precision ln10    *
3543 0CFC6 AF2          C=0    W
3544 0CFC9 2C           P=     12
3545 0CFCB 3348         LCHEX  5684          (low digits = 5684018- )
           65
3546 0CFD1 AF7          D=C    W
3547 0CFD4 7AB1         GOSUB  LNC10+        + LN10
3548 0CFD8 CE           C=C-1  A             (C,D)=2.3025 85092 99404 / 56840..0
3549 0CFDA AFD          BCEX   W
3550 0CFDD D2           C=0    A
3551                                         A=...X_high...
3552                                         B=0230258509299404
3553                                         C=...X_low..00000
3554                                         D=568400....00000
3555
3556                                         R0=0--------xponX
3557 0CFDF 25           P=     5
```

Now, how to make ALL these extra digits visible in a Basic expression?
PEEKing the nibbles from the ROM would be cheating, no?

J-F

EMAIL PM WWW FIND QUOTE REPORT

05-05-2018, 07:53 PM **Post: #4**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **J-F Garnier Wrote:** ⇨ (05-05-2018 05:44 PM)
>
> PEEKing the nibbles from the ROM would be cheating, no?

Thanks for your interest, J-F.

I'll comment extensively on each Step when I post my original solutions next week but as for your question, simply abide by the specified Requirements for each Step. For instance, if it doesn't specify that you can't pray for a miracle, then you can (and hope for the best) ... :-)

Best regards and have a nice weekend.
V.
.

PM FIND EDIT QUOTE REPORT

05-05-2018, 08:54 PM **Post: #5**

**rprosperi**
Senior Member

Posts: 3,278
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ⇨ (05-05-2018 07:53 PM)
>
> ...simply *abide by the specified Requirements for* ***each Step***. For instance, if it doesn't specify that you can't pray for a miracle, then you can (and hope for the best) ... :-)

PEEK/POKE is prohibited for some steps, *but not for steps 1 & 2*.

That said, praying may also be in order here, as the hint implies... 

---

**--Bob Prosperi**

EMAIL PM FIND QUOTE REPORT

05-08-2018, 06:16 PM **Post: #6**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ➡ (05-04-2018 11:39 PM)
>
> Step the Third:
> ... you don't have to write a program but instead solve right from the command line the following equation ...
> ...
> - you can execute more than one command line in succession if need be
> - you *can't run, call or use* any program code whatsoever

The requirements put strong constraints on a possible solution:
the command line is limited to 96 characters,
FNROOT is not useable here (unless I missed something...),
it is very difficult (or impossible) to make a loop from the keyboard command line and include tests in it.
So I chose a dichotomic search (with a trick to avoid a test in the loop).
It's easy to find the maximum number of dichotomic steps needed, so no need of a termination test.

Here is my solution in two command lines and about 100 characters:
the first line initializes an array with two values defining an interval where the root is for sure:

```
DESTROY A @ A(0)=-1 @ A(1)=49
```

the second line drives a basic dichotomic search:

```
FOR J=1 TO 46 @ X=(A(0)+A(1))/2 @ S=0 @ FOR I=1 TO 100 @ S=S+SQR(X+I)-7 @ NEXT I @ A(S>0)=X @ NEXT J
```

(spaces added only for readability, don't type them in order to fit in a 96-char line).

The result is in X:

```
>X
3.28838856026
```

Any better/shorter/clever solution?

J-F

EMAIL PM WWW FIND QUOTE REPORT

05-10-2018, 12:53 AM **Post: #7**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

**Hi all**, and **hi J-F**:

To all, just a brief reminder that there's still 24 hours left to submit any further solutions to some or all of the *6 Steps* of this challenge before I post my original solutions tomorrow, as stated in my original post.

As I did in previous challenges, I'll post my original solutions and comments only for those *Steps* for which code for one or more solutions (or attempted solutions) has been posted by the readers. The ones which receive no inputs will best be left for future use.

To **J-F**:

> **J-F Garnier Wrote:**
>
> Here is my solution in two command lines and about 100 characters:

The first time I tried it, it gave me a *"Subscript"* error because your first line assigns a value to *A(0)* and I had *OPTION BASE 1* set at the time.

This must be addressed, which is why I (nearly) always include *DESTROY ALL* as the first statement in any program or command line, followed by *OPTION BASE* if I use arrays. Adding those two statements to your first command line makes your solution about 15 characters longer or so. No big deal.

Your final *>X* to display the computed root can be included at the end of your second command line, it still fits, so you don't need a third mini-command-line just for it.

> **J-F Garnier Wrote:**
>
> It is very difficult (or impossible) to make a loop from the keyboard command line and include tests in it.

It's difficult but not impossible, it can be done as long as it all fits in a 96-char line. For instance, to search for and display the very *first* 4-digit square that includes *"444"*, you could issue a command line like this:

```
>FOR X=32 TO 99 @ X=X+100*(POS(STR$(X*X),"444")#0) @ NEXT X @ DISP X-101;RES^2

      38      1444
```

As you can see, it doesn't loop all the way to $99^2$ but *stops* looping at the first square which fulfills the test condition and displays both the number and its square.

In a similar fashion, to search for the very first 4-digit *"square plus one"* that happens to be a *prime* number, this command line would do it:

```
>FOR X=32 TO 99 @ X=X+100*(PRIM(X*X+1)=0) @ NEXT X @ DISP X-101,RES^2+1

      36      1297
```

This last example uses the *PRIM* function from the *JPC ROM*, which also includes structures that can be used to implement this functionality. For instance, the last example could be rewritten like this:

```
>X=31 @ REPEAT @ X=X+1 @ UNTIL PRIM(X*X+1)=0 OR X>99 @ DISP X,X*X+1

      36      1297
```

which works as well except for the fact that if no X value would produce a prime then the last value (100) would be output instead , while the first version would output "-1", perhaps better indicating the *not-found* condition.

Best regards.
V.

PM FIND EDIT QUOTE REPORT

05-10-2018, 10:23 AM (This post was last modified: 05-10-2018 10:28 AM by J-F Garnier.) **Post: #8**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

**Valentin Albillo Wrote:** (05-10-2018 12:53 AM)

> **J-F Garnier Wrote:**
> It is very difficult (or impossible) to make a loop from the keyboard command line and include tests in it.
>
> It's difficult but not impossible, it can be done as long as it all fits in a 96-char line. ...

Thanks for your comments, Valentin. What I meant is that it is not possible to include a IF THEN structure in a FOR NEXT loop, because NEXT is not allowed after THEN or ELSE.
So we have to find other ways to take decisions as I did in my solution and as you illustrated with your interesting examples.

For the OPTION BASE issue, here is an updated solution (I don't like changing global settings like OPTION BASE):

```
>DESTROY ALL @ A(1)=-1 @ A(2)=49
>FOR J=1 TO 46 @ X=(A(1)+A(2))/2 @ S=0 @ FOR I=1 TO 100 @ S=S+SQR(X+I)-7 @ NEXT I @ A(1+(S>0))=X @ NEXT J
>X
3.28838856026
```

Waiting for your own solution(s) !

J-F

EMAIL PM WWW FIND QUOTE REPORT

05-10-2018, 11:55 PM **Post: #9**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**[VA] Short & Sweet Math Challenges #23 - My Solutions & Comments**

**Hi, all:**

Time to wrap up this **S&SMC#23**. Sadly, it seems it failed to grasp the attention of forum members judging by the scarcity of solutions/comments posted. I naively thought that there would be a considerable number of **HP-71B** and **Star Wars** fans here that would welcome this homage to the unofficial *Star Wars' Day, May the 4th*, but only **J-F Garnier** took notice and contributed his always valuable and interesting solutions and comments.

Also, although the very last *Step (Step the Sixth)* can be solved with every HP calc model out there from the *HP-10C* upwards (at least for 4-5 digits if not all 10-12), no one posted anything on it, not even *J-F*, a real pity. Anyway, these are my original solutions plus assorted comments:

## Step the First:

*"Write a program which accepts from the user an integer N from 1 to 19 and outputs the Nth digit of Log(10), the natural logarithm of 10 (= 2.302585092994045684 to 19-digit accuracy)."*

My original solution is this 1-line program (48 bytes):

```
1 DISP (PEEK$("0CFCD",4)&"4"&PEEK$("0D19A",14))[20-VAL(DISP$)][1,1]
```

Let's see:

```
> 1 [RUN]       2
> 2 [RUN]       3
>18 [RUN]       8
>19 [RUN]       4
```

The *HP-71B* 64 Kb ROM which holds the entire operating system and *BASIC* language includes a number of explicit mathematical constants to a high precision and we can make use of that fact to obtain the pre-stored constants without the need to compute them anew, thus saving lots of time. It's just a question of knowing their address within the ROM.

My solution uses **PEEK** to assemble the 19-digit value of *Log(10)* from the ROM, in reverse order, and then uses the *string-slicing* operator applied *twice* in succession not to a string variable but directly to the resulting string expression to obtain the digits from the one specified by the user to the end of the string, and once more to isolated the precise digit. The digit position specified by the user isn't stored in a variable (via *INPUT*, for instance) but taken directly from the display via **DISP$**. This fulfills the *no-variables* requirement.

*Lessons learnt:*

- the use of **PEEK$** to extract mathematical constants from the **HP-71B** *System* ROM.
- the use of **DISP$** to obtain a value from the user without storing it in a variable
- the use of the *string-slicing* operator **[]** applied directly to a string expression
- the fact that the [] operator can be applied *more than once* in succession

## Step the Second:

*"Write a program which accepts from the user an integer N from 1 to 32 and outputs the Nth digit of Pi/2 (= 1.5707963267948966192313216916397 to 32-digit accuracy)."*

My original solution is this 3-line program (152 bytes):

```
1 DESTROY ALL @ X=33-VAL(DISP$) @ A$=PEEK$("0DB0F",16)&PEEK$("0DA91",15) @ C=-1
2 FOR I=1 TO 31 @ N=VAL(A$[I,I]) @ N$=STR$(N+N) @ IF LEN(N$)=1 THEN N$="0"&N$
3 A$[I,I]=STR$(VAL(N$[2])+C) @ C=VAL(N$[1,1]) @ NEXT I @ DISP (A$&STR$(C))[X,X]
```

Let's see:

```
> 1 [RUN]       1
```

```
> 2 [RUN]        5
>31 [RUN]        9
>32 [RUN]        7
```

On the surface this seems exactly like the previous *Step*, only with another constant instead of *Log(10)*. The difference is that while the 19-digit value of *Log(10)* does appear in the ROM, the 32-digit value of *Pi/2* does *not*.

However, the value of *Pi/4* does so it's simply a matter of retrieving it from the ROM and multiply it times 2 within the requirements, in particular *not* using standard math functions and *no* arithmetical operator except **+** or **-**, which my solution does digit-by-digit. Once assembled, the required digit at the location specified by the user is retrieved and output.

*Lessons learnt:*

- some constants not in the ROM can be easily derived from the ones available there

## Step the Third:

*"Solve right from the command line the following equation: Sqrt(x+1) + Sqrt(x+2) + Sqrt(x+3) + ... + Sqrt(x+98) + Sqrt(x+99) + Sqrt(x+100) = 700"*

My original solution is this 2-line command-line expression (117 characters in all):

```
>DESTROY ALL @ DIM T$[1400] @ FOR I=1 TO 100 @ T$=T$&"+SQR(FVAR+"&STR$(I)&")" @ NEXT I
>VAL("FNROOT(0,0,"&T$[2]&"-700)")

      3.28838856035 (correct 12-digit value: 3.28838856020)
```

The first command line dimensions a string to hold the full equation, which is then assembled into it in a simple loop, and then the second line simply completes the following expression which uses the **FNROOT** root from the *Math* ROM to solve the equation with identical initial guesses, **0** and **0**. Finally, the entire expression is evaluated using **VAL**, immediately obtaining the root, *3.28838856035*.

The assembled expression which *VAL* evaluates is this 1307-character string:

```
FNROOT(0,0,SQR(FVAR+1)+SQR(FVAR+2)+SQR(FVAR+3)+SQR(FVAR+4)+SQR(FVAR+5)+SQR(F
VAR+6)+SQR(FVAR+7)+SQR(FVAR+8)+SQR(FVAR+9)+SQR(FVAR+10)+SQR(FVAR+11)+SQR(FVAR
+12)+SQR(FVAR+13)+SQR(FVAR+14)+SQR(FVAR+15)+SQR(FVAR+16)+SQR(FVAR+17)+SQR(FVA
R+18)+SQR(FVAR+19)+SQR(FVAR+20)+SQR(FVAR+21)+SQR(FVAR+22)+SQR(FVAR+23)+SQR(FV
AR+24)+SQR(FVAR+25)+SQR(FVAR+26)+SQR(FVAR+27)+SQR(FVAR+28)+SQR(FVAR+29)+SQR(F
```

```
VAR+30)+SQR(FVAR+31)+SQR(FVAR+32)+SQR(FVAR+33)+SQR(FVAR+34)+SQR(FVAR+35)+SQR(
FVAR+36)+SQR(FVAR+37)+SQR(FVAR+38)+SQR(FVAR+39)+SQR(FVAR+40)+SQR(FVAR+41)+SQR
(FVAR+42)+SQR(FVAR+43)+SQR(FVAR+44)+SQR(FVAR+45)+SQR(FVAR+46)+SQR(FVAR+47)+SQ
R(FVAR+48)+SQR(FVAR+49)+SQR(FVAR+50)+SQR(FVAR+51)+SQR(FVAR+52)+SQR(FVAR+53)+S
QR(FVAR+54)+SQR(FVAR+55)+SQR(FVAR+56)+SQR(FVAR+57)+SQR(FVAR+58)+SQR(FVAR+59)+
SQR(FVAR+60)+SQR(FVAR+61)+SQR(FVAR+62)+SQR(FVAR+63)+SQR(FVAR+64)+SQR(FVAR+65)
+SQR(FVAR+66)+SQR(FVAR+67)+SQR(FVAR+68)+SQR(FVAR+69)+SQR(FVAR+70)+SQR(FVAR+71
)+SQR(FVAR+72)+SQR(FVAR+73)+SQR(FVAR+74)+SQR(FVAR+75)+SQR(FVAR+76)+SQR(FVAR+7
7)+SQR(FVAR+78)+SQR(FVAR+79)+SQR(FVAR+80)+SQR(FVAR+81)+SQR(FVAR+82)+SQR(FVAR+
83)+SQR(FVAR+84)+SQR(FVAR+85)+SQR(FVAR+86)+SQR(FVAR+87)+SQR(FVAR+88)+SQR(FVAR
+89)+SQR(FVAR+90)+SQR(FVAR+91)+SQR(FVAR+92)+SQR(FVAR+93)+SQR(FVAR+94)+SQR(FVA
R+95)+SQR(FVAR+96)+SQR(FVAR+97)+SQR(FVAR+98)+SQR(FVAR+99)+SQR(FVAR+100)-700)
```

and as you can see, both *FNROOT* and VAL *have* no problem in dealing with expressions of *any length* whatsoever, they're limited only by available RAM, *not* by the 96-character limit of the command-line itself. The same is true of *INTEGRAL*.

An alternative for the second command line would be the following:

```
>FNROOT(0,0,VAL(T$[2])-700)
```

which is a slightly shorter (just 6 characters less) but *much slower*. Matter of fact the first version is *2.2 times faster*, the reason being that the first expression parses the long expression to evaluate only *once*, then has *FNROOT* working with the parsed expression, while the second version has *FNROOT* executing *VAL* repeatedly as many times as needed to find and refine the root and thus parsing the long string *multiple* times.

**J-F** also managed to find a nice solution within the requirements (except for the fact that he uses initial guesses different from 0, which was also a requirement but that would be nitpicking) but as he uses a user-code loop within a loop it is much slower than using the assembly-language *FNROOT* and *VAL* applied to an expression parsed once into fast internal format. Anyway, congratulations to *J-F* for his very clever way to achieving the stated goal.

Just for fun and to demonstrate this *no-limits* capability even further, solving the following equation:

$$\text{Sqrt}(x+1) + \text{Sqrt}(x+2) + \text{Sqrt}(x+3) + \ldots + \text{Sqrt}(x+98) + \text{Sqrt}(x+99) + \text{Sqrt}(x+200) = 2018$$

entails executing these command lines (118 character in all):

```
>DESTROY ALL @ DIM T$[2800] @ FOR I=1 TO 200 @ T$=T$&"+SQR(FVAR+"&STR$(I)&")" @ NEXT I
>VAL("FNROOT(0,0,"&T$[2]&"-2018)")
```

```
10.4122270141
```
*(correct 12-digit value: 10.4122270160)*

and this time the expression which *VAL* evaluates is the 2708-character string:

```
FNROOT(0,0,SQR(FVAR+1)+SQR(FVAR+2)+SQR(FVAR+3)+SQR(FVAR+4)+SQR(FVAR+5)+SQR(F
VAR+6)+SQR(FVAR+7)+SQR(FVAR+8)+SQR(FVAR+9)+SQR(FVAR+10)+SQR(FVAR+11)+SQR(FVAR
+12)+SQR(FVAR+13)+SQR(FVAR+14)+SQR(FVAR+15)+SQR(FVAR+16)+SQR(FVAR+17)+SQR(FVA
R+18)+SQR(FVAR+19)+SQR(FVAR+20)+SQR(FVAR+21)+SQR(FVAR+22)+SQR(FVAR+23)+SQR(FV
AR+24)+SQR(FVAR+25)+SQR(FVAR+26)+SQR(FVAR+27)+SQR(FVAR+28)+SQR(FVAR+29)+SQR(F
VAR+30)+SQR(FVAR+31)+SQR(FVAR+32)+SQR(FVAR+33)+SQR(FVAR+34)+SQR(FVAR+35)+SQR(
FVAR+36)+SQR(FVAR+37)+SQR(FVAR+38)+SQR(FVAR+39)+SQR(FVAR+40)+SQR(FVAR+41)+SQR
(FVAR+42)+SQR(FVAR+43)+SQR(FVAR+44)+SQR(FVAR+45)+SQR(FVAR+46)+SQR(FVAR+47)+SQ
R(FVAR+48)+SQR(FVAR+49)+SQR(FVAR+50)+SQR(FVAR+51)+SQR(FVAR+52)+SQR(FVAR+53)+S
QR(FVAR+54)+SQR(FVAR+55)+SQR(FVAR+56)+SQR(FVAR+57)+SQR(FVAR+58)+SQR(FVAR+59)+
SQR(FVAR+60)+SQR(FVAR+61)+SQR(FVAR+62)+SQR(FVAR+63)+SQR(FVAR+64)+SQR(FVAR+65)
+SQR(FVAR+66)+SQR(FVAR+67)+SQR(FVAR+68)+SQR(FVAR+69)+SQR(FVAR+70)+SQR(FVAR+71
)+SQR(FVAR+72)+SQR(FVAR+73)+SQR(FVAR+74)+SQR(FVAR+75)+SQR(FVAR+76)+SQR(FVAR+7
7)+SQR(FVAR+78)+SQR(FVAR+79)+SQR(FVAR+80)+SQR(FVAR+81)+SQR(FVAR+82)+SQR(FVAR+
83)+SQR(FVAR+84)+SQR(FVAR+85)+SQR(FVAR+86)+SQR(FVAR+87)+SQR(FVAR+88)+SQR(FVAR
+89)+SQR(FVAR+90)+SQR(FVAR+91)+SQR(FVAR+92)+SQR(FVAR+93)+SQR(FVAR+94)+SQR(FVA
R+95)+SQR(FVAR+96)+SQR(FVAR+97)+SQR(FVAR+98)+SQR(FVAR+99)+SQR(FVAR+100)+SQR(F
VAR+101)+SQR(FVAR+102)+SQR(FVAR+103)+SQR(FVAR+104)+SQR(FVAR+105)+SQR(FVAR+106
)+SQR(FVAR+107)+SQR(FVAR+108)+SQR(FVAR+109)+SQR(FVAR+110)+SQR(FVAR+111)+SQR(F
VAR+112)+SQR(FVAR+113)+SQR(FVAR+114)+SQR(FVAR+115)+SQR(FVAR+116)+SQR(FVAR+117
)+SQR(FVAR+118)+SQR(FVAR+119)+SQR(FVAR+120)+SQR(FVAR+121)+SQR(FVAR+122)+SQR(F
VAR+123)+SQR(FVAR+124)+SQR(FVAR+125)+SQR(FVAR+126)+SQR(FVAR+127)+SQR(FVAR+128
)+SQR(FVAR+129)+SQR(FVAR+130)+SQR(FVAR+131)+SQR(FVAR+132)+SQR(FVAR+133)+SQR(F
VAR+134)+SQR(FVAR+135)+SQR(FVAR+136)+SQR(FVAR+137)+SQR(FVAR+138)+SQR(FVAR+139
)+SQR(FVAR+140)+SQR(FVAR+141)+SQR(FVAR+142)+SQR(FVAR+143)+SQR(FVAR+144)+SQR(F
VAR+145)+SQR(FVAR+146)+SQR(FVAR+147)+SQR(FVAR+148)+SQR(FVAR+149)+SQR(FVAR+150
)+SQR(FVAR+151)+SQR(FVAR+152)+SQR(FVAR+153)+SQR(FVAR+154)+SQR(FVAR+155)+SQR(F
VAR+156)+SQR(FVAR+157)+SQR(FVAR+158)+SQR(FVAR+159)+SQR(FVAR+160)+SQR(FVAR+161
)+SQR(FVAR+162)+SQR(FVAR+163)+SQR(FVAR+164)+SQR(FVAR+165)+SQR(FVAR+166)+SQR(F
VAR+167)+SQR(FVAR+168)+SQR(FVAR+169)+SQR(FVAR+170)+SQR(FVAR+171)+SQR(FVAR+172
)+SQR(FVAR+173)+SQR(FVAR+174)+SQR(FVAR+175)+SQR(FVAR+176)+SQR(FVAR+177)+SQR(F
VAR+178)+SQR(FVAR+179)+SQR(FVAR+180)+SQR(FVAR+181)+SQR(FVAR+182)+SQR(FVAR+183
)+SQR(FVAR+184)+SQR(FVAR+185)+SQR(FVAR+186)+SQR(FVAR+187)+SQR(FVAR+188)+SQR(F
VAR+189)+SQR(FVAR+190)+SQR(FVAR+191)+SQR(FVAR+192)+SQR(FVAR+193)+SQR(FVAR+194
)+SQR(FVAR+195)+SQR(FVAR+196)+SQR(FVAR+197)+SQR(FVAR+198)+SQR(FVAR+199)+SQR(F
VAR+200)-2018)
```

which is quite a sight to behold.

*Lessons learnt:*

- That **VAL** and **FNROOT** (and **INTEGRAL**) can deal with expressions *limited only by available RAM*, not by the 96-character limit of the command line.

## Step the Fourth:

*"Write a program which accepts a positive integer N from the user and outputs both the number and its square for every value from N down to 0, both included, one pair per line."*

As **J-F** wrote in his nice solution, the task would be utterly trivial were it not for the *no-variables* requirement. My original solution is this 2-line (46 bytes) program:

```
1 DISP USING "#,^";(VAL(DISP$)+1)^2
2 DISP SQR(RES)-1,RES^2 @ IF SQR(RES) THEN 2
```

Let's see:

```
>15 [RUN]

 15       225
 14       196
 13       169
 ...

 3        9
 2        4
 1        1
 0        0
```

The trick to avoid using variables is first of all to accept a value from the user *directly from the command line* using **DISP$**, instead of inputting it to a variable, and then to store and handle it using **RES**, a system location which stores the value of the most recently evaluated **or displayed** numeric expression (whether real- or complex-valued but not string-valued).

Thus, the first line simply puts in *RES* an adequate initial value for the loop in the second line by simply displaying it. Normally this would result in this spurious initial value being displayed too, as a side effect, but this is avoided by the **USING** *image* which supresses both the value and the return carriage.

The second line then enters a simple loop where *SQR(RES)-1* retrieves and displays the next value to square and *RES^2* squares it and displays the result as well. As soon as *SQR(RES)* reaches 0, the loop (and the program) ends. Notice that *IF* does **not** update the value of *RES* to the expression being evaluated and tested.

By the way, there's another 1-line version which is 49 bytes instead of 46:

```
1 DISP USING "#,^";(VAL(DISP$)+1)^2 @ 'A': DISP SQR(RES)-1,RES^2 @ IF SQR(RES) THEN 'A'
```

It simply uses a *local label 'A' midline* to implement the loop instead of a line number, thus avoiding the use of a second line.

*Lessons learnt:*

- that **RES** can be used to occasionally replace a variable
- that **USING** a *image* we can store a value in *RES* without actually displaying it
- that a *local label midline* allows looping within part of a single line

## Step the Fifth:

*"Write a program which accepts from the user a single-digit Id, then accepts from the user a text to scan for said Id and output the name associated with that Id.*

*The format of the text to scan (up to 80 characters long, say) is as follows:*

*(Id1):(Name1),(Id2):(Name2), ... , (IdN):(NameN)*

*where (Id) is a single digit and (Name) is a string of up to 30 characters A-Z & spaces. The Id aren't necessarily in numerical order in the text, but the Id sought for must appear somewhere within the text."*

This is similar in spirit to *Step 4* above but with the added difficulty of having to handle *two* inputs (the *Id* to search for and the *text* where to search for it) without using variables. My original solution is a 2-line program (74 bytes) but as no one posted any code or comment for this *Step*, it will be reserved for possible use at some future time.

## Step the Sixth:

*"We'll call "Selfie" to any positive N-digit integer number which has the property that if you sum its N digits raised to the Nth power you get the original number backwards. For instance, the 7-digit number 5271471 is a Selfie:*

*5271471 => 5^7 + 2^7 + 7^7 + 1^7 + 4^7 + 7^7 + 1^7 = 1741725, which is 5271471 backwards*

*Write a program to find all Selfies from 1 to 9 digits long (for 10-digit HP calcs, 29 in all) or from 1 to 11 digits long (for 12-digit HP calcs, 37 in all). 0 is not a positive number so it's not a Selfie."*

Same here, no one posted any code or comments for this one either so I'll also save my original solution for a future article or something. As

I said at the beginning of this post, it's a real pity as this Step was solvable (partially or in full) using most any HP calc model, and there are at least 3 ways to handle it, including brute force which will quickly become unfeasible as for 11-digit numbers there would be some 90 billion numbers to try, so more sophisticated techniques are required to reach that far in reasonable times (a few minutes).

By the way, I didn't ask for 12/10-digit solutions because for some 12/10-digit numbers the sum of their digits raised to the 12th/10th power exceeds the 12/10-digit range. Also, *there are no 12-digit Selfies* (but there's a *unique* 10-digit one).

That's all for now.
Regards.
V.

PM FIND EDIT QUOTE REPORT

05-11-2018, 12:56 AM **Post: #10**

**rprosperi**
Senior Member

Posts: 3,278
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)
>
> **Hi, all:**
>
> Time to wrap up this **S&SMC#23**.

Thanks for these painstakingly detailed notes about your solutions; these are always enjoyable even I've not had time, or sometimes the skills, to tackle them. I particularly like your use of "Lessons learned" to highlight the key techniques used for each solution, this helps a lot to drive these points in.

> **Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)
>
> - the fact that the [] operator can be applied *more than once* in succession

I had no idea one could apply the substring operator (I like 'slicer' better) multiple times. Though it appears to not be documented it does make sense following the 71b's syntax, etc. Very clever!

Thanks for another set of interesting and educational challenges. I suspect all the drama taking place in parallel distracted folks from playing along.

**--Bob Prosperi**

EMAIL PM FIND QUOTE REPORT

05-11-2018, 08:29 PM **Post: #11**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

**Hi, Bob !**

> **rprosperi Wrote:** ⇨ (05-11-2018 12:56 AM)
> Thanks for these painstakingly detailed notes about your solutions; these are always enjoyable even I've not had time, or sometimes the skills, to tackle them. I particularly like your use of "Lessons learned" to highlight the key techniques used for each solution, this helps a lot to drive these points in.

Certainly these challenges I concoct *do* take a lot of time, first for getting a workable idea, then to find my own solutions to it, and finally to write, proofread and format the looong posts (the one for the challenge proper and the one for the solutions).

I recently added the feature you mention and I'm glad you like it. Though not a teacher by trade, I'm pretty fond of sharing knowledge with others, which can be considered kind of "teaching".

> **Quote:**
> I had no idea one could apply the substring operator (I like 'slicer' better) multiple times. Though it appears to not be documented it does make sense following the 71b's syntax, etc. Very clever!

I knew about the "slicer" many decades ago, back in the very early 80's, where I saw it available in **HP-85**'s BASIC. There, it could be applied only to string variables or elements of a string array but not to string expressions, and only once. The **HP-71B**'s BASIC version is much enhanced over the *HP-85*'s one and can be applied also to string expressions and any number of times in sequence.

In the first challenge I use it to save bytes, because `[20-VAL(DISP$)][1,1]` is shorter than other ways of achieving the same result with a single slicing operation.

> **Quote:**

> Thanks for another set of interesting and educational challenges. I suspect all the drama taking place in parallel distracted folks from playing along.

Thanks to you for your continued appreciation and for letting me know, it means a lot to me to ascertain that my efforts didn't go to waste.

Have a nice weekend and best regards.
V.

PM FIND EDIT QUOTE REPORT

05-11-2018, 09:51 PM **Post: #12**

**pier4r**
Senior Member

Posts: 1,962
Joined: Nov 2014

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)
>
> I naively thought that there would be a considerable number of **HP-71B** and **Star Wars** fans here

I think that some challenges requires just more time as one has to have the right conditions to attack them. Especially if the challenges are focused on few systems.

Also recently the forum went through some turbolences so this may have distracted the community as well.

In any case, always kudos for the extensive explanation!

---

Wikis are great, Contribute :)

EMAIL PM FIND QUOTE REPORT

05-11-2018, 09:57 PM **Post: #13**

**Egan Ford**
Member

Posts: 165
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)

no one posted any code or comments for this one either so I'll also save my original solution for a future article or something. As I said at the beginning of this post, it's a real pity as this Step was solvable (partially or in full) using most any HP calc model, and there are at least 3 ways to handle it, including brute force which will quickly become unfeasible as for 11-digit numbers there would be some 90 billion numbers to try, so more sophisticated techniques are required to reach that far in reasonable times (a few minutes).

By the way, I didn't ask for 12/10-digit solutions because for some 12/10-digit numbers the sum of their digits raised to the 12th/10th power exceeds the 12/10-digit range. Also, *there are no 12-digit Selfies* (but there's a *unique* 10-digit one).

I put 3 hours into this and one sleepless night (could not stop thinking how to optimize). It'll continue to haunt me just like SSMC #20's last problem continues to do today. Coincidently I was reading Henry Ibstedt yesterday.

EMAIL PM FIND QUOTE REPORT

05-12-2018, 10:37 AM **Post: #14**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

Valentin, your solution of the Step the Third is really great!

I didn't known that it was possible to evaluate long expressions in that way.
I learnt something today (yes, it's still possible to learn new things, after more than 30 years of HP71 usage), thanks again:

> **Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)
>
> *Lessons learnt:*
>
> - That **VAL** and **FNROOT** (and **INTEGRAL**) can deal with expressions *limited only by available RAM*, not by the 96-character limit of the command line.

How did you come to discover this possibility? By chance or rational exploration?

Of course, it is only possible thanks to the VAL function of the HP71, that is actually an expression evaluation function, rather than the classic Basic VAL function that can only convert a number from its ASCII representation.
For instance, the solution is not applicable to the HP75.

The HP71 design team did a great job at the time.

J-F

EMAIL PM WWW FIND QUOTE REPORT

05-13-2018, 01:59 AM **Post: #15**

**cortopar**
Junior Member

Posts: 15
Joined: May 2018

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

Valentin,

I bet it doesn't feel like it, but even if this thread only had two posts, your original challenge and your answers, it would be tremendously valuable to many of us who observe more than interact.

Your posts over the years are about 98% of the reason for my interest in the 71b.

Thanks for continuing to carry the 71b torch, and thanks for all your posts on other models that have added to my RPN and math knowledge.

All the best,
Bob

PM FIND QUOTE REPORT

05-14-2018, 01:15 AM **Post: #16**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

.
**Hi**, **J-F Garnier** and **cortopar**:

> **J-F Garnier Wrote:** ⇨ (05-12-2018 10:37 AM)
> Valentin, your solution of the Step the Third is really great!

Thank you very much, I'm glad you liked it.

**Quote:**

I didn't known that it was possible to evaluate long expressions in that way.
I learnt something today (yes, it's still possible to learn new things, after more than 30 years of HP71 usage), thanks again:

You're welcome. The fact that you learned something new is proof enough that you're stilll pretty young, you know what they say about old dogs not being able to learn new tricks. :-D

**Quote:**

How did you come to discover this possibility? By chance or rational exploration?

I discovered that **VAL** can evaluate long expressions soon after I got me a pretty (and expensive!) **HP-71B** complete with **HP-IL** ROM, **Forth/Assembler** ROM and all the **IDS** volumes.

Afterwards, when I also got the **Math** ROM back inf 1985 (without its IDS, regrettably) I did test the limits of most of its functions and empirically discovered that the *funny functions* **FNROOT** and **INTEGRAL** were akin to *VAL* in that regard. I took notice of those facts at the time and then it was just a matter of finding the right occasion to share the knowledge.

**Quote:**

Of course, it is only possible thanks to the VAL function of the HP71, that is actually an expression evaluation function, rather than the classic Basic VAL function that can only convert a number from its ASCII representation. For instance, the solution is not applicable to the HP75.

Indeed, I've found no other version of *VAL* (or its equivalent in other languages) which can do that. The *HP-85/86/87* couldn't either, neither could the *HP9816/26/36* or the various versions of *Pascal* or *Visual This/Visual That*, etc.

**Quote:**

The HP71 design team did a great job at the time.

They did 95% great things and a few % not that great. For instance, I will never forgive the *<expletive>* who decided the inclusion of that abomination called *"CALC mode"*, which utterly wasted 5 Kb of the 64 Kb ROM which could've been put to much, much better use.

Or the *<milder expletive>* who implemented *string handling* in such a way that the system must have the whole string on the stack to do anything with it: say you need to extract or change the Nth character of a long string, the operating system cannot simply do it using pointer/address manipulations, no, it must copy the whole string to the stack, using lots of valuable RAM and time and thus making using long

strings inefficient and slow. And there are more (meager string functions, missing complex functions, poor *PEEK/POKE* implementation, missing matrix functions in the *Math* ROM, etc) ...

> **cortopar Wrote:**
> I bet it doesn't feel like it, but even if this thread only had two posts, your original challenge and your answers, it would be tremendously valuable to many of us who observe more than interact.

Thank you very much, **Bob**, you're far too kind and I'm really glad you appreciate my *S&SMCs*.

> **Quote:**
> Your posts over the years are about 98% of the reason for my interest in the 71b. Thanks for continuing to carry the 71b torch, and thanks for all your posts on other models that have added to my RPN and math knowledge.

Thanks again. Though I tend to use the *HP-71B* more than other models, for obvious reasons, I *do like* and admire a lot the classic RPN ones and I have scores of programs written by myself for the *HP-11C, HP-15C, HP-25, HP-34C, HP-67/97, HP-41C, HP42S* and even the *HP35s*, most of them unpublished.

I intend to scan or type them anew as soon as I can find some way to put them online, which I'm actually looking into right now.

Best regards.
V.
.

PM FIND EDIT QUOTE REPORT

05-14-2018, 03:59 PM **Post: #17**

**John Keith**
Senior Member

Posts: 389
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Valentin Albillo Wrote:** ⇨ (05-14-2018 01:15 AM)
> Indeed, I've found no other version of *VAL* (or its equivalent in other languages) which can do that. The *HP-85/86/87* couldn't either, neither could the *HP9816/26/36* or the various versions of *Pascal* or *Visual This/Visual That*, etc.

Except, of course, for RPL and other LISP-derived languages.

**Quote:**

They did 95% great things and a few % not that great. For instance, I will never forgive the *<expletive>* who decided the inclusion of that abomination called *"CALC mode"*, which utterly wasted 5 Kb of the 64 Kb ROM which could've been put to much, much better use.

Amen! They could at least have given us an RPN calculator mode. HP-41 compatible ideally, although I don't know if that would have fit into 5K.

Thanks from me also for a fun and invigorating challenge, although mostly "above my pay grade".

John

EMAIL PM FIND QUOTE REPORT

05-15-2018, 12:21 AM **Post: #18**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

.
Hi, **John**:

**Quote:**

Amen! They could at least have given us an RPN calculator mode. HP-41 compatible ideally, although I don't know if that would have fit into 5K.

It probably would, 5 Kb of optimized Saturn assembly language can reach very far. Just consider the many worthwhile functionalities of the basic HP-71B and it all fits in just 59 Kb, even still including some space/time-wasting garbage.

Also, the HP-71B operating system already includes essential RPN functionalities. Each numeric expression entered is parsed and tokenized into internal RPN form for storage as program code and/or execution, then decompiled as necessary for program listings, editing and debugging. It would simply be a matter of harnessing that existing functionality and exposing it to the user.

**Quote:**

Thanks from me also for a fun and invigorating challenge, although mostly "above my pay grade".

You're welcome, I'm truly glad that you liked it and greatly appreciate your kind feedback.

Regards.
V.
.

PM FIND EDIT QUOTE REPORT

05-15-2018, 07:22 PM **Post: #19**

**Jeff O.**
Member

Posts: 166
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

**Valentin Albillo Wrote:** ⇨ (05-10-2018 11:55 PM)

**Hi, all:**

Time to wrap up this **S&SMC#23**. ...

*"We'll call "Selfie" to any positive N-digit integer number which has the property that if you sum its N digits raised to the Nth power you get the original number backwards. For instance, the 7-digit number 5271471 is a Selfie:*

*5271471 => 5^7 + 2^7 + 7^7 + 1^7 + 4^7 + 7^7 + 1^7 = 1741725, which is 5271471 backwards*

*Write a program to find all Selfies from 1 to 9 digits long (for 10-digit HP calcs, 29 in all) or from 1 to 11 digits long (for 12-digit HP calcs, 37 in all). 0 is not a positive number so it's not a Selfie."*

Same here, no one posted any code or comments for this one either so I'll also save my original solution for a future article or something. As I said at the beginning of this post, it's a real pity as this Step was solvable (partially or in full) using most any HP calc model...

I'll confess to not having read through your challenge closely enough to see your note that this particular step might be potentially solvable by other models. I think the 71B is a wonderful machine and am glad to have an example, but have never attempted to master its use, so I assumed this challenge was not for me. Upon further review, Step the 6th is the kind of number manipulation challenge that I have enjoyed attempting on various models, along the lines of some of the HHC programming contests. With that said, reading that Egan put 3 hours and one sleepless night into it, and it will continue to haunt him, kind of scares me off a bit. My usual inclination with such problems is to just go

ahead and try for a brute force method, then try to optimize. The DM42 is fast, I would like to see how many digits it could handle in a reasonable time by brute force, so if I get the time I may have a go at it.

In any case, thanks for your challenges, please don't be put off by a lower than hoped-for response. Next time I'll be sure to read through more carefully!

---

*Dave - My mind is going - I can feel it.*

PM FIND QUOTE REPORT

05-15-2018, 07:42 PM **Post: #20**

**Maximilian Hohmann**
Senior Member

Posts: 557
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenges #23: "May the 4th Be With You !" ...**

> **Jeff O. Wrote:** ⇨ (05-15-2018 07:22 PM)
> In any case, thanks for your challenges, please don't be put off by a lower than hoped-for response.

Same here! Although I do have some 71Bs and even a Math ROM I am only superficially familiar with it's many functions and would not have been able to solve a single one of these challenges. Especially the ones which require PEEKing the digits of maththematical constants out of the ROM... (The last time I wrote the word "PEEK" before this reply must have been ca. 1983 when I did some machine language programming on my Sinclair ZX81). Nontheless these challenges and the answers are a pleasure to read and think about!

EMAIL PM FIND QUOTE REPORT

« Next Oldest | Next Newest »

Enter Keywords Search Thread

**Pages (3):** 1 2 3 Next »

NEW REPLY

View a Printable Version
Send this Thread to a Friend
Subscribe to this thread

User(s) browsing this thread: Valentin Albillo*