## [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special



04-04-2018, 09:08 AM (This post was last modified: 04-04-2018 09:29 AM by Didier Lachieze.) **Post: #21**

**Didier Lachieze**
Senior Member

Posts: 1,117
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **J-F Garnier Wrote:** ⇒ (04-02-2018 10:53 PM)
> After the desserts, back to the main course!
>
> I have a nice little HP71 program that is able to output the (assumed) correct result for each of the six test numbers.
> However, I will not publish my very clever program here, just the results as NO or YES:

Thanks J-F for not spoiling the solution.

This morning I solved the main course with some lateral thinking and a little help from Parzival. It was a pure moment of joy to find the solution!

I have now a 37-byte program in my 42S which provides the result for each of the 6 test numbers. I will publish it tomorrow (April 5 by 8PM CET) to leave some more time for others to find the solution.

Thanks again Valentin, you made my day. This is a marvelously crafted challenge

EMAIL PM FIND QUOTE REPORT

04-05-2018, 09:01 AM **Post: #22**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **Gerson W. Barbosa Wrote:** ⇒ (04-04-2018 04:38 AM)
> `OVF*OVF*RAD(UNF)` is another variation. But Dessert #2 won't be over until -Pie/100 and -Pie/10000 are served.

Still room for the last pieces of dessert?
RAD(OVF*OVF%UNF) -> -Pi/100
RAD(OVF%OVF%UNF) -> -Pi/10000

J-F

EMAIL PM WWW FIND QUOTE REPORT

04-05-2018, 02:35 PM (This post was last modified: 04-06-2018 06:01 PM by Gerson W. Barbosa.) **Post: #23**

**Gerson W. Barbosa**
Senior Member

Posts: 1,135
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **J-F Garnier Wrote:** ⇒ (04-05-2018 09:01 AM)
> Still room for the last pieces of dessert?
> RAD(OVF*OVF%UNF) -> -Pi/100
> RAD(OVF%OVF%UNF) -> -Pi/10000

Very nice!

Skipped the main course and took just a tiny bit of pie. That's what I call being on a diet :-)
Thank you all for the HP-71B tricks!

Gerson.

Edited for Grammar.

EMAIL PM FIND QUOTE REPORT

04-05-2018, 08:06 PM **Post: #24**

**Didier Lachieze**
Senior Member

Posts: 1,117
Joined: Dec 2013

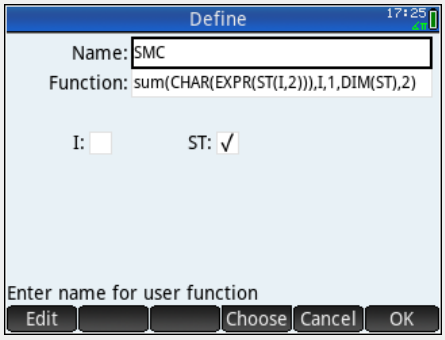
**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

It took me quite a long time to find the solution to the main course. With my limited knowledge of primality tests I was going nowhere until I took a step back, looked at the overall challenge and started to think to the date of the challenge posting, which led me to Parzival's Easter Eggs hunt in "Ready Player One", the latest Spielberg movie. At that point I thought: "maybe Valentin has cleverly placed some clues to the solution in the numbers themselves", so I took a different look at the test numbers and bingo, each test number was an Easter egg !

Here is my 42s program for the main course. I managed to squeeze out one byte since yesterday, for a total of 36 bytes.
Usage: XEQ "P", enter the test number, press R/S and see the result.

```
00 { 36-Byte Prgm }
01▸LBL "P"
02 CLA
03 AON
04 PROMPT
05 AOFF
06 ALENG
07▸LBL 00
08 ATOX
09 10
10 ×
11 ATOX
12 +
13 528
14 -
15 XTOA
16 R↓
17 2
18 -
19 X>0?
20 GTO 00
21 AVIEW
22 END
```

Here is also a 48-character user function for the HP Prime: `sum(CHAR(EXPR(ST(I,2))),I,1,DIM(ST),2)`




Usage: provide the input number as a string, e.g. SMC("8082737769637879")

Note: you need first to create the variable ST (for ex. with ST:="") before defining the user function.

EMAIL PM FIND QUOTE REPORT

04-05-2018, 08:16 PM **Post: #25**

**J-F Garnier**
Senior Member

Posts: 302
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **Didier Lachieze Wrote:** ⇒ (04-05-2018 08:06 PM)
> Here is my 42s program for the main course. ..

The program can even run on a 41 (with x-functions) for the three shortest test numbers (<=24 digits).
With Free42, you can directly paste the numbers into the ALPHA register.
Watch the results!

Thanks to Valentin for this nice puzzle !

J-F

EMAIL PM WWW FIND QUOTE REPORT

04-06-2018, 06:08 AM **Post: #26**

**Valentin Albillo**
Senior Member

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

**[VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special - My Solutions**

**Hi all,**

As always, thank you very much for the high degree of participation in my ***SSMC#22 April 1st 2018 Spring Special*** and most importantly the high quality of your various inputs, whether individual or *"as a team"* (I like the concept !), which indeed managed to find the correct solutions to all parts of it, sometimes actually producing my exact original solution and at other times producing an equivalent variation thereof.

These are my original solutions plus assorted comments:

## Main Course:

This is my original version for the **HP-71B**, an UDF (User-Defined Function, 2 lines, 72 bytes) which accepts as argument the number whose primality or compositeness we want to know (as a string, to cater with inputs more than 12 digits long) and outputs the number's answer to that question.

```
1 DEF FNS$(N$) @ S$="" @ FOR I=1 TO LEN(N$) STEP 2
2 S$=S$&CHR$(VAL(N$[I,I+1])) @ NEXT I @ FNS$=S$ @ END DEF
```

Speaking of which, there are any number of powerful algorithms to check a number for primality but I think that my approach is quite novel, namely:

> ***"Why not ask the number itself if it's prime or composite ? Surely it should know !"***.

The above UDF implements just that approach. Let's see how it fares with the six test numbers:

```
>FNS$("8082737769637879")

      PRIME?NO                              {correct, composite divisible by 5701}

>FNS$("8969837365776578736769808273776 9")

      YESIAMANICEPRIME                      {correct, nice or not it's indeed a prime}

>FNS$("67797780798373846978857766698 2")

      COMPOSITENUMBER                       {obviously composite}

>FNS$("7365778082737769847979")

      IAMPRIMETOO                           {correct, it's a prime}

>FNS$("67797780798373846965838769767 6")

      COMPOSITEASWELL                       {obviously composite}

>FNS$("7378686969688082737769")

      INDEEDPRIME                           {correct, it's a prime}
```

Of course these are not the only numbers who truthfully answer when asked, there are *billions and billions* which will obligue as well, for instance:

```
FNS$("83797769328082737769")              -> SOME PRIME         { correct, it's a prime }
FNS$("667371328082737769")                -> BIG PRIME          { ditto }
FNS$("65768379328082737769")              -> ALSO PRIME         { ditto }
FNS$("7339776580827377 6949484837")       -> I'MAPRIME100%      { ditto }
FNS$("736577787984808273776983798 28289") -> IAMNOTPRIMESORRY   { composite, divisible by 43^2 }
FNS$("677977807983738469")                -> COMPOSITE          { ditto, divisible by 79 }
FNS$("787984677977807983738469 33333333") -> NOTCOMPOSITE!!!!   { correct, it's an enthusiastic prime }
FNS$("91808273776993")                    -> [PRIME]            { correct, it's a prime }
FNS$("787932837989328082737779")          -> NO SOY PRIMO       { Spanish composite, div. by 2663 }
FNS$("80823771696332787933")              -> PRIME? NO!         { composite, divisible by 3 }
FNS$("687386738373667669668951")          -> DIVISIBLEBY3       { ditto }
```

Other numbers do care to answer but it seems they're not that sure about their own status:

```
FNS$("7879843283858269")                  -> NOT SURE           { composite, divisible by 9601 }
FNS$("87738372327332757869 87")           -> WISH I KNEW        { ditto , divisible by 7669 }
FNS$("89798532846976763277 69")           -> YOU TELL ME        { ditto , divisible by 3 }
FNS$("877279326765826983 63")             -> WHO CARES?         { ditto , divisible by 3 }
FNS$("6669658483327769")                  -> BEATS ME           { ditto , divisible by 17 }
```

Also, still others do not even care but even seem to *resent* being asked and reply rudely:

```
FNS$("7669658669327769326576797869")      -> LEAVE ME ALONE     { composite, divisible by 3 }
FNS$("7179326587658933")                  -> GO AWAY!           { ditto, divisible by 367651 }
FNS$("6669658432738433")                  -> BEAT IT!           { ditto, divisible by 11 }
FNS$("837279794432837279 79")             -> SHOO, SHOO         { ditto, divisible by 3 }
```

And finally, the *worst* offenders of all, some numbers do reply but only to ***lie*** *shamelessly* through their teeth !

```
FNS$("65787984726982677977807983738469")  -> ANOTHERCOMPOSITE   { such liar ! ... you're a prime ! }
```

To be honest, the vast majority seem to be under the influence or something because they reply with *gibberish* when asked but I won't give any examples here as they're quite common and so pretty easy to find.

All in all, I'd say my groundbreaking, novel primality check it's a **great success**, don't you think ? ... XD

Also, this is my RPN version for the **HP42S** (a program, 21 steps, 39 bytes, no numbered registers or variables)

```
      00 { 39-Byte Prgm }
      01 LBL "P?"
      02 "N?"
      03 AON
      04 PROMPT
      05 ALENG
      06 2
      07 /
      08 LBL 00
      09 ATOX
      10 10
      11 X
      12 ATOX
      13 +
      14 528
      15 -
      16 XTOA
      17 Rv {roll down}
      18 DSE ST X
      19 GTO 00
      20 AVIEW
      21 END

XEQ "P?"

      N?
            8082737769637879 [R/S] -> PRIME?NO
XEQ "P?"

      N?
            896983736577657873676980827377 69 [R/S] -> YESIAMANICEPRIME
```

etc.

## Dessert 1:

Shortest is (also it uses no digits, strings, or functions):

```
      1 DISP MAXREAL*EPS (8 bytes, 8-5 = 3 bytes for the expression itself)
```

Other less efficient possibilities that people might try:

```
      1 DISP 9.99999999999       (14 bytes, 9 bytes for the expression)
      1 DISP 10-1E-11            (12 bytes, 7 bytes for the expression)
      1 DISP RAD(DEG(-INX))-OVF  (12 bytes, 7 bytes for the expression, also no digits)
      1 DISP NEIGHBOR(10,0)      (12 bytes, 7 bytes for the expression)
      1 DISP 10/3*3              (11 bytes, 6 bytes for the expression)
      1 DISP MAXREAL/1E499       (10 bytes, 5 bytes for the expression)
      1 DISP RAD(DEG(6))+4       (10 bytes, 5 bytes for the expression)
```

## Dessert 2:

Shortest are (all of them 6 bytes):

```
      >RAD(OVF*UNF*OVF)

            -3.14159265359 {-Pi}

      >RAD(OVF*UNF%OVF)

            -3.14159265359E-2 {-Pi/100}

      >RAD(OVF%UNF%OVF)

            -3.14159265359E-4 {-Pi/10000}
```

and their various permutations. As I said, **RAD** is *not* a trigonometric *function*, as it's merely multiplication by a conversion factor and so it's perfectly legal, while **ANGLE** *is* a trigonometric function (a variant of the arctangent function) and so *not* legal for this challenge.

## Dessert 3:

Some people offered valid F(X) and G(X) but the original I had in mind is the simple pair **TANH(X)** and **ATANH(X)**. For these functions we have:

**- For the HP-71B:**

```
            X      ATANH(TANH(X))    % Error
      ----------------------------------------
      10.000000      10.000037      0.000373 %
      11.000000      10.999905     -0.000867 %
      14.000000      14.162084      1.157744 %
      14.100000      14.162084      0.440313 %
      14.200000      14.162084     -0.267013 %
      14.300000      14.162084     -0.964447 %
      14.400000      14.162084     -1.652193 %
```

```
14.500000      14.162084      2.330454 % <<< exceeds 2% absolute error
```

The *smallest value* for which % Error is greater than 2% in absolute value can be found this way:

```
>FNROOT(10,14.5,ABS(100*(ATANH(TANH(FVAR))/FVAR-1))-2)

      14.4511062736
```

which is the correct smallest value as the relative error is:

```
>100*(ATANH(TANH(14.4511062736))-14.4511062736)/14.4511062736

      -1.9999999995 (%)
```

i.e.: ~ 2% error, as required:

- **For the HP-11C/HP-15C** and other 10-digit calcs featuring hyperbolic functions, we may use this simple routine to explore the % error:

```
01 LBL A
02 ENTER
03 TANH
04 ATANH
05 D% {delta %}
06 ABS
07 RTN

[USER] [FIX 4]

12 [A] -> 1.1708 (%)
12.1 [A] -> 1.9876 (%)
12.2 [A] -> 2.7910 (%) <<< exceeds 2% absolute error
```

The *smallest value* for which *% Error* is greater than 2% in absolute value can be easily found using an **HP-15C** by solving this litte program (a variation of the above code):

```
01 LBL A
02 ENTER
03 TANH
04 ATANH
05 D% {delta %}
06 ABS
07 2
08 -
09 RTN

[USER] [FIX 9]

12 [ENTER] 12.2 [SOLVE A] -> 12.10152965
```

- let's check:

```
[ENTER] [TANH] [ATANH] [D%] -> -1.999999975
```

~ 2% error, as required

- **For the HP42S**, this code allows for exploration. It's the same code as the one for the 10-digit **HP-11C**, say, but the results are the same as those for the 12-digit **HP-71B**:

```
01 LBL A
02 ENTER
03 TANH
04 ATANH
05 %CH
06 ABS
07 END

12 XEQ A -> 0.0274 (%)
14 XEQ A -> 1.1577 (%)
14.5 XEQ A -> 2.3305 (%) <<< exceeds 2%
```

etc.

## Dessert 4:

Shortest is:

```
>INX-INX;INX-UNF;INX-OVF;INX-DVZ;-INX;-UNF;-OVF;-DVZ;-IVL;-INX-UNF;-INX-OVF (74-char)

      0 1 2 3 4 5 6 7 8 9 10
```

and there are zillions of permutations and variations. For instance, you can get 0 by INX-INF, UNF-UNF, ..., EPS-EPS, etc. and you can get 1 by INX/INX, UNF/UNF, ..., EPS/EPS, ... and so on and so forth. None of them are less than 74-char long and I don't think a shorter solution is possible (though I'd love to be proved wrong).

Finally, as a free bonus, I'll give the factorization I discovered for the big number I gave in the prologue to the challenge, namely:

```
55555555555555444444444444444433333333333333332222222222222222211111111111111 =

      3063441154048486369668261625739 * 181350163955772670068231705843686316121284149
```

where both factors are prime, of course. To check it using my HP-71B and a very simple multiprecision multiplication UDF, just execute this:

```
>FNM$("3063441154048486369668261625739","181350163955772670068231705843686316121284149")

      55555555555555444444444444444433333333333333332222222222222222211111111111111
```

which checks Ok.

That's all. Thanks for your interest and really glad you liked it. See you in ***S&SMC#23*** ! :-)

Regards.
V.

PM FIND EDIT QUOTE REPORT

04-06-2018, 07:45 AM **Post: #27**

**pier4r**
Senior Member

Posts: 1,962
Joined: Nov 2014

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

Let the numbers talk is a really nice idea!

Wikis are great, Contribute :)

EMAIL PM FIND QUOTE REPORT

04-06-2018, 04:11 PM **Post: #28**

**John Keith**
Senior Member

Posts: 389
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **Didier Lachieze Wrote:** ⇒ (04-03-2018 12:13 AM)
> Nice solutions, and I'm learning new HP-71 tricks with this challenge.

Myself as well, although I haven't used my 71B in a long time. Thank you Valentin for an interesting and fun challenge!

Though it seems like cheating, this HP49/50 program seems to meet all the rules for the main course, and it's hard to beat for size at 15.5 bytes:

<< ISPRIME? >>

John

EMAIL PM FIND QUOTE REPORT

04-06-2018, 04:35 PM **Post: #29**

**Gene**
Moderator

Posts: 880
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

Interesting challenge, Valentin!

Couple of questions on the Main Course.

1) First, how about a deeper explanation? Inquiring minds want to know.

2) The 42S version does not work for a lot of numbers, but does others, as you suggest.

Have you in a sneaky manner pre-chosen the numbers such that the approach can spit out meaningful alpha?

Don't get me wrong, that is perhaps a GREATER achievement if so. :-)

I would just like to understand better what is going on. Thanks as always...

EMAIL PM FIND QUOTE REPORT

04-06-2018, 05:52 PM (This post was last modified: 04-06-2018 06:52 PM by Gerson W. Barbosa.) **Post: #30**

**Gerson W. Barbosa**
Senior Member

Posts: 1,135
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **Gene Wrote:** ⇒ (04-06-2018 04:35 PM)
> I would just like to understand better what is going on.

The Dec and Char columns in this ASCII table can be used to hand-decode the messages.

# ASCII Table

| Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | 32 | 20 | 40 | [space] | 64 | 40 | 100 | @ | 96 | 60 | 140 | ` |
| 1 | 1 | 1 | | 33 | 21 | 41 | ! | 65 | 41 | 101 | A | 97 | 61 | 141 | a |
| 2 | 2 | 2 | | 34 | 22 | 42 | " | 66 | 42 | 102 | B | 98 | 62 | 142 | b |
| 3 | 3 | 3 | | 35 | 23 | 43 | # | 67 | 43 | 103 | C | 99 | 63 | 143 | c |
| 4 | 4 | 4 | | 36 | 24 | 44 | $ | 68 | 44 | 104 | D | 100 | 64 | 144 | d |
| 5 | 5 | 5 | | 37 | 25 | 45 | % | 69 | 45 | 105 | E | 101 | 65 | 145 | e |
| 6 | 6 | 6 | | 38 | 26 | 46 | & | 70 | 46 | 106 | F | 102 | 66 | 146 | f |
| 7 | 7 | 7 | | 39 | 27 | 47 | ' | 71 | 47 | 107 | G | 103 | 67 | 147 | g |
| 8 | 8 | 10 | | 40 | 28 | 50 | ( | 72 | 48 | 110 | H | 104 | 68 | 150 | h |
| 9 | 9 | 11 | | 41 | 29 | 51 | ) | 73 | 49 | 111 | I | 105 | 69 | 151 | i |
| 10 | A | 12 | | 42 | 2A | 52 | * | 74 | 4A | 112 | J | 106 | 6A | 152 | j |
| 11 | B | 13 | | 43 | 2B | 53 | + | 75 | 4B | 113 | K | 107 | 6B | 153 | k |
| 12 | C | 14 | | 44 | 2C | 54 | , | 76 | 4C | 114 | L | 108 | 6C | 154 | l |
| 13 | D | 15 | | 45 | 2D | 55 | - | 77 | 4D | 115 | M | 109 | 6D | 155 | m |
| 14 | E | 16 | | 46 | 2E | 56 | . | 78 | 4E | 116 | N | 110 | 6E | 156 | n |
| 15 | F | 17 | | 47 | 2F | 57 | / | 79 | 4F | 117 | O | 111 | 6F | 157 | o |
| 16 | 10 | 20 | | 48 | 30 | 60 | 0 | 80 | 50 | 120 | P | 112 | 70 | 160 | p |
| 17 | 11 | 21 | | 49 | 31 | 61 | 1 | 81 | 51 | 121 | Q | 113 | 71 | 161 | q |
| 18 | 12 | 22 | | 50 | 32 | 62 | 2 | 82 | 52 | 122 | R | 114 | 72 | 162 | r |
| 19 | 13 | 23 | | 51 | 33 | 63 | 3 | 83 | 53 | 123 | S | 115 | 73 | 163 | s |
| 20 | 14 | 24 | | 52 | 34 | 64 | 4 | 84 | 54 | 124 | T | 116 | 74 | 164 | t |
| 21 | 15 | 25 | | 53 | 35 | 65 | 5 | 85 | 55 | 125 | U | 117 | 75 | 165 | u |
| 22 | 16 | 26 | | 54 | 36 | 66 | 6 | 86 | 56 | 126 | V | 118 | 76 | 166 | v |
| 23 | 17 | 27 | | 55 | 37 | 67 | 7 | 87 | 57 | 127 | W | 119 | 77 | 167 | w |
| 24 | 18 | 30 | | 56 | 38 | 70 | 8 | 88 | 58 | 130 | X | 120 | 78 | 170 | x |
| 25 | 19 | 31 | | 57 | 39 | 71 | 9 | 89 | 59 | 131 | Y | 121 | 79 | 171 | y |
| 26 | 1A | 32 | | 58 | 3A | 72 | : | 90 | 5A | 132 | Z | 122 | 7A | 172 | z |
| 27 | 1B | 33 | | 59 | 3B | 73 | ; | 91 | 5B | 133 | [ | 123 | 7B | 173 | { |
| 28 | 1C | 34 | | 60 | 3C | 74 | < | 92 | 5C | 134 | \ | 124 | 7C | 174 | \| |
| 29 | 1D | 35 | | 61 | 3D | 75 | = | 93 | 5D | 135 | ] | 125 | 7D | 175 | } |
| 30 | 1E | 36 | | 62 | 3E | 76 | > | 94 | 5E | 136 | ^ | 126 | 7E | 176 | ~ |
| 31 | 1F | 37 | | 63 | 3F | 77 | ? | 95 | 5F | 137 | _ | 127 | 7F | 177 | |

I've manually encoded three messages, but none have worked. It should not be too difficult, however, to find meaningful numbers using a program that combines both an encoder and a primality tester. That's what Valentin probably has used.

EMAIL PM FIND QUOTE REPORT

04-06-2018, 08:12 PM **Post: #31**

**rprosperi**
Senior Member

Posts: 3,278
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

**Gene Wrote:** ⇨ (04-06-2018 04:35 PM)

I would just like to understand better what is going on. Thanks as always...

Boy, are you gonna wince hard with a loud DOH!

**--Bob Prosperi**

EMAIL PM FIND QUOTE REPORT

04-06-2018, 08:54 PM **Post: #32**

**Gene**
Moderator

Posts: 880
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

lol. No, I saw the character value manipulation going on, which is also of course why some values such as our favorite 10 digit largest prime 9999 9999 67 spit out only nonsense. :-)

I'm just curious if Valentin really waded in backward from various text outputs and really constructed the original numbers that way. I suppose so, but wanted to ask.

EMAIL PM FIND QUOTE REPORT

04-06-2018, 09:32 PM **Post: #33**

**rprosperi**
Senior Member

Posts: 3,278
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

**Valentin Albillo Wrote:** ⇨ (04-06-2018 06:08 AM)

**Hi all**,

As always, thank you very much for the high degree of participation in my ***SSMC#22 April 1st 2018 Spring Special*** and most importantly the high quality of your various inputs, whether individual or *"as a team"* (I like the concept !), which indeed managed to find the correct solutions to all parts of it, sometimes actually producing my exact original solution and at other times producing an equivalent variation thereof.

These are my original solutions plus assorted comments:

Thanks for the Easter treat with your SSMC#22 Valentin, as always very educational and this time even more fun than usual.

For the main course, I only got as far as noticing the examples all had an even number of digits. If I noticed that this was listed on April 1st, that may have moved me closer to figuring it out, but unlike Didier, with whom I share having no background in factoring Primes, I very quickly moved on to desert.

The conditions for Desert #2 pointed me quickly to the many unique Functions in the 71, but as you saw, that took a team effort to wrestle down.

Desert #4's solution is also quite interesting and satisfying; I had just started to explore some of these, having (re-)learned the nature of these flag functions on Desert #2, but didn't get far before JFG's brilliant reply.

A question for you true 71B masters: I thought all function calls (similar to '41 XROM) were 2 bytes long: 1) the LEX ID and 2) the particular Fn in that LEX (each ranging up to 255, so needing a full byte).

Yet, the answer to D#2, "RAD(OVF*UNF*OVF)" is clearly 4 functions but only 6 bytes. Hmph?!

I don't want to derail this too far from the main topic, but since 'size counts' I thought it may be relevant to ask here.

**--Bob Prosperi**

EMAIL PM FIND QUOTE REPORT

04-07-2018, 03:05 AM **Post: #34**

**Valentin Albillo**
Senior Member

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

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

.
**Hi all**,

Thanks a lot for the extra feedback, much appreciated (actually your feedback it's the fuel that energizes me to post challenges and other materials !). As I did three days ago I'll address here the various things you recently either commented or asked. Let's begin:

**pier4r Wrote:**
Let the numbers talk is a really nice idea!

Thanks ! As far as I know, it's a novel concept, I've never seen it before.

**Mike (Stgt) Wrote:**
Thank you for this flock of easter eggs! In addition I learned that the HP11C does hyperbolic functions. Alas my question if ->H and the inverse ->H.MS was valid functions twosome stays unanswered.

The **HP-11C** does indeed hyperbolics (very useful to solve one-real-root cubic equations, by the way). And yes, your twosome **->H** and **->H.MS** are indeed perfectly valid solutions for ***Dessert 3*** (or equivalently **HR** and **HMS** functions in the **HP-71B**). I didn't mention them specifically in my solutions but I did say *"Some people offered valid F(X) and G(X)"*, which included them of course. Congratulations for finding them, good lateral thinking.

**John Keith Wrote:**
Thank you Valentin for an interesting and fun challenge!
Though it seems like cheating, this HP49/50 program seems to meet all the rules for the main course, and it's hard to beat for size at 15.5 bytes:

<< ISPRIME? >>

Thanks for your appreciation and kind comments. As for the `<< ISPRIME? >>` program I concur that it's quite short but, as they say, *"the proof of the pudding is in the eating"* so: What results does it give when applied to the six test numbers I gave ?

**Gene Wrote:**
Interesting challenge, Valentin!
Couple of questions on the Main Course.

1) First, how about a deeper explanation? Inquiring minds want to know.
[...]
I would just like to understand better what is going on. Thanks as always...

Thanks for your continued appreciation, Gene, but my explaining it all would ruin the magic and in fact if I told you I'd have to kill you. :-D

**rprosperi Wrote:**
Thanks for the Easter treat with your SSMC#22 Valentin, as always very educational and this time even more fun than usual.

Thak you very much, I'm happy to know that you find them educational and fun, that's my goal. I learned a lot while having lots of fun while reading Martin Gardner's *Mathematical Recreations* series of books and since then I've always thought that having fun inmensely enhances learning.

> **Quote:**
> The conditions for Desert #2 pointed me quickly to the many unique Functions in the 71, but as you saw, that took a team effort to wrestle down.

Hehe, as I said, I like the concept of solving challenges *as a team*, well done.

> **Quote:**
> A question for you true 71B masters: I thought all function calls (similar to '41 XROM) were 2 bytes long: 1) the LEX ID and 2) the particular Fn in that LEX (each ranging up to 255, so needing a full byte).
>
> Yet, the answer to D#2, "RAD(OVF*UNF*OVF)" is clearly 4 functions but only 6 bytes. Hmph?!

Well, **RAD(OVF*UNF*OVF)** if one single-parameter function (**RAD**), three parameterless functions (i.e.: "constants"), **OVF**, **UNF**, **OVF**, and two arithmetic operators (*****). Each of them is 1-byte so 1+3+2 = 6 bytes.

As for all function calls being 2-byte, that's not the case. There are 1-byte functions and there are 4-byte functions, etc. For instance:

**1 DISP** is 5 bytes
1 DISP **1** is 6 bytes (the **1** is 1-byte)
1 DISP **LOG**(1) is 7 bytes (**LOG** is 1-byte)
1 DISP **LN**(1) is 7 bytes (**LN**, an alternate spelling of **LOG**, is also 1-byte)
1 DISP **LOG10**(1) is 7 bytes (**LOG10** is also 1-byte)
1 DISP **LGT**(1) is 10 bytes (**LGT**, an alternate spelling of **LOG10**, is 4 bytes)
1 DISP **LOGP1**(1) is 10 bytes (**LOGP1** is also 4 bytes)
1 DISP **LOG2**(1) is 10 bytes (**LOG2** is in the **Math ROM** and it's also 4 bytes)

so you see, in the above examples there are mainframe 1-byte functions, mainframe 4-byte functions, and external 4-byte functions. The mainframe 4-byte functions are the ones *less used* (**LOG10** and **LGT** are the identical function but the former is 1-byte while the latter is 4-byte).

Something similar happens with the exponential functions, **EXP** is 1-byte while **EXPM1** is 4-bytes. As for functions in the **Math ROM** we also have that **DET** is 4-byte and **DET(A)** is 5-byte (one extra byte for the **A**). **FOUR** is also 4-byte (self-describing, it seems), **TRN** is 4-byte as well, ditto for **SYS**, and so on (these last three also need a **1 MAT A=**, which is 8-byte in itself, plus the bytes needed by their argument(s)

I could go on but you get the idea and it's quite easy to either experiment using **CAT** or a little routine using **PEEK$** to display the length and tokenization of an arbitrary program line.

Again, thanks to all and best regards.
V.
.

PM FIND EDIT QUOTE REPORT

04-07-2018, 04:32 AM **Post: #35**

**rprosperi**
Senior Member

Posts: 3,278
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

> **Valentin Albillo Wrote:** ⇨ (04-07-2018 03:05 AM)
> As for all function calls being 2-byte, that's not the case. There are 1-byte functions and there are 4-byte functions, etc. For instance:
>
> **1 DISP** is 5 bytes
> 1 DISP **1** is 6 bytes (the **1** is 1-byte)
> 1 DISP **LOG**(1) is 7 bytes (**LOG** is 1-byte)
> 1 DISP **LN**(1) is 7 bytes (**LN**, an alternate spelling of **LOG**, is also 1-byte)
> 1 DISP **LOG10**(1) is 7 bytes (**LOG10** is also 1-byte)
> 1 DISP **LGT**(1) is 10 bytes (**LGT**, an alternate spelling of **LOG10**, is 4 bytes)
> 1 DISP **LOGP1**(1) is 10 bytes (**LOGP1** is also 4 bytes)
> 1 DISP **LOG2**(1) is 10 bytes (**LOG2** is in the **Math ROM** and it's also 4 bytes)
>
> so you see, in the above examples there are mainframe 1-byte functions, mainframe 4-byte functions, and external 4-byte functions. The mainframe 4-byte functions are the ones *less used* (**LOG10** and **LGT** are the identical function but the former is 1-byte while the latter is 4-byte).

Thanks for clarifying, I have been exploring this exact thing today, editing line 1, using CAT to check, edit again, CAT again, etc. Very illuminating! I was getting what I thought were truly odd results, but your well-chosen examples show the storage needed for varying function statements is more subtle and complex than I recalled; alternate names requiring different bytes is unexpected too.

As for alternate names, presumably included for familiarity to different users, while **LN** for **LOG** makes sense, **LOGT** for **LOG10** seems rather odd; is this commonly used in Math research, I never encountered it in Engineering.

**--Bob Prosperi**

EMAIL PM FIND QUOTE REPORT

04-07-2018, 12:59 PM **Post: #36**

**Ángel Martin**

Posts: 940

Senior Member Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

**Valentin Albillo Wrote:** ⇒ (04-04-2018 02:08 AM)

> **Mike (Stgt) Wrote:**
> "At one's ease"? So the following determinations are not too serious? Please explain.
> ...
> "We may say", why not "we say"?

Please, Mike (Stgt), English is not my native language and I'm doing what I can with it so please cut me some slack, will you ?

The use of "may" in this very sentence is completely appropriate even if you mean it as a certainty. In fact it's a resource used by many native speakers as well , so Valentin is right on target also on the linguistic side.

The error made frequently (usually by non-native speakers) is to interpret language in a literal manner - which it is not, unless of course you're writing a legal text or a technical procedure (and even there it's flexible).

FWIW, this doesn't add value to the contents of the thread (which is a highly enjoyable contribution), but splitting hairs is a nice pastime for some.

PM FIND QUOTE REPORT

04-07-2018, 08:02 PM **Post: #37**

**John Keith**
Senior Member

Posts: 389
Joined: Dec 2013

**RE: [VA] Short & Sweet Math Challenge #22: April 1st, 2018 Spring Special**

**Valentin Albillo Wrote:** ⇒ (04-07-2018 03:05 AM)

> **John Keith Wrote:**
> Thank you Valentin for an interesting and fun challenge!
> Though it seems like cheating, this HP49/50 program seems to meet all the rules for the main course, and it's hard to beat for size at 15.5 bytes:
>
> << ISPRIME? >>

Thanks for your appreciation and kind comments. As for the `<< ISPRIME? >>` program I concur that it's quite short but, as they say, *"the proof of the pudding is in the eating"* so: What results does it give when applied to the six test numbers I gave ?

.

It returns 1 for numbers that are prime, and 0 for numbers that are composite. The results are correct for all 6 numbers in your challenge.

I meant my "solution" to be facetious since ISPRIME? is a built-in function. I did write a couple of ASCII-based programs but they weren't very impressive (87 bytes, 59.5 bytes if external libraries are allowed).

John

EMAIL PM FIND QUOTE REPORT

« Next Oldest | Next Newest »

Enter Keywords Search Thread

**Pages (2):** « Previous 1 2

NEW REPLY

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

User(s) browsing this thread: Valentin Albillo*