[VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier" Threaded Mode | Linear Mode

03-21-2019, 03:08 AM Post: #1

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

[VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, all ! Happy Spring to all of you !

... and once again to commemorate its arrival, here you are, a brand new ***Short & Sweet Math Challenge #24 "Spring Special 5-tier"*** to give you all a chance to put to good use both your favourite **HP calculator** and your programming ingenuity (*NOT* your *Google Search* proficiency). Try all ***5 tiers*** below and see what's your top one !

Rules:

- Any **HP calc** of your choice may be used but I'll suggest a *Minimum Recommended Model* (**MRM**) for each tier, which is the simplest model I deem capable of solving it more or less comfortably.
- Using anything other than a **physical** or **emulated HP calculator** is strictly disallowed. Also **no** *VBA*, *Excel*. *Pascal*, *C/C#/C++*, *Java*, *Python*, *Haskell*, etc. code, please go elsewhere for that. You must write your code in a language supported in some HP calc (i.e.: ***RPN***, ***RPL***, ***71B BASIC***/***FORTH***, etc).
- Googling the solutions is lame beyond belief and, frankly, if you do you'll be but a *sore loser* in my eyes :).

Tier 1: Noob

*[MRM: **HP-11C** and up]*

Let's begin with something affordable. As it happens, I have **2** HP-**71**B and a **41**C in my collection and I love them dearly so I'll pay them a little homage here: we'll call ***Homage Number*** to any *10-digit* positive integers which are *multiples* of **271**, *divisible* by **41** and further their digits are *all distinct*.

The Challenge:

Write a program that takes no inputs but simply finds out and outputs just how many ***Homage Numbers*** there are. No need to output any of them, just count'em and do it fast !

Your code should be as *fast* and *short* as possible, in that order. I'll post my original code (a 2-liner) and results for the **HP-71B** .

Tier 2: Beginner

*[MRM: **HP-29C** and up]*

Consider the function $\mathbf{S_B(N)}$ which returns the *sum* of the *base-**B*** digits of an integer **N**. For instance:

$\mathbf{S_2}(2019) = S_2(11111100011_2) = 1+1+1+1+1+1+0+0+0+1+1 = \mathbf{8}$
$\mathbf{S_5}(2019) = S_5(31034_5) = 3+1+0+3+4 = \mathbf{11}$
$\mathbf{S_{10}}(2019) = 2+0+1+9 = \mathbf{12}$
$\mathbf{S_{16}}(2019) = S_{16}(7E3_{16}) = 7+E\ (=14)\ +3 = \mathbf{24}$
$\mathbf{S_{36}}(2019) = S_5(1K3_{36}) = 1+K\ (=20)\ +3 = \mathbf{24}$

The Challenge:

Write a program that accepts a base **B** *(2 to 36)* and outputs in order those ***prime*** numbers **N** such that $\mathbf{S_B(N)}$ is **composite** and *distinct* from the previous ones.
For instance, this is what your program should generate for bases **B = 6**, **8** and **16** *(hexadecimal)*:

B = 6: *11, 19, 23, ... 179, ...*
B = 8: *11, 13, 23, ... 191, ...*
B = 16: *19, 23, 29, ... 223, ...*

Once verified that your code reproduces the above sample results, go on and generate the corresponding sequences for base **B = 31** first and then for base **B = 7** .
What results do you get ? Which is the *smallest* (first) prime in each sequence ? *How many* elements can you generate for each sequence ? What about other bases ?

Again, your code should be as *fast* and *short* as possible, in that order. I'll post my original code (a 6-liner) and results for the **HP-71B** .

Tier 3: Intermediate

*[MRM: **HP-25** and up]*

Consider the real numbers **77.4019...** and **231.4859...** , which are sums of *distinct* non-negative integer powers of ***e*** *(= exp(1) = 2.71828...):*

$$77.4019... = e^1 + e^3 + e^4$$
$$231.4859... = e^0 + e^2 + e^3 + e^4 + e^5.$$

Those positive real numbers that are either powers of ***e*** or sums of distinct powers of ***e*** form an increasing sequence whose first term is **1** (i.e.: e^0) and matter of fact we have that **77.4019..** is the **26**th term in the sequence and **231.4859...** is the **61**th term.

The Challenge:

Generalizing to powers of an arbitrary real number ***P*** >= ***e***, write a program or function which accepts as input both ***P*** and an index ***k*** and returns the corresponding ***k***th term in the sequence (in the example above we would have *MyFunction(**e**, 26) = 77.4019...* and *MyFunction(**e**, 61) = 231.4859...*). Your code should be as short and fast as possible.

Now use your program/function to find the **1,000,000**th term and the **3,141,593**th term when ***P = e*** as well as the **1,234,567**th term and the **2,718,282**th term when ***P = Pi***. Also, just for show, use it to list the first 10 terms or so for each sequence.

I'll post both a *1-line* user-defined function for the **HP-71B** and an equivalent *24-step RPN* program for the **HP-25** (which should work with little or no change in *all RPN*-based HP calcs).

Tier 4: Advanced

*[MRM: **HP-11C** and up]*

Consider the ***n***-point dataset (x_i, y_i) where $x_i = 1, 2, 3, 4, 5, 6, ..., n$ (the natural numbers) and $y_i = 2, 3, 5, 7, 11, 13, ..., p_n$ (the *prime* numbers), and the *(n-1)st degree polinomial fit* to this dataset of the form:

$$P(x) = a_0 + a_1 (x-1) + a_2 (x-1) (x-2) + ... + a_{n-1} (x-1) (x-2) (x-3) ... (x-(n-1))$$

The Challenge:

Write a program that takes no inputs but computes and outputs the limit of the sum of the coefficients $a_0, a_1, ..., a_{n-1}$ when ***n*** tends to *infinity*. Your program must be as short and fast as possible and must compute the limit to the *10-12 digits* maximum accuracy of your calc, give or take a few *ulps*.

I'll post a *4-line*, *168-byte* program for the **HP-71B** which computes and outputs the limit in ~0.2 sec *(Emu71)* but a fast *RPN* version for the *HP-11C* and up is also perfectly possible.

Tier 5: Guru

*[MRM: **HP-11C** and up]*

Surely you're well aware of the elementary trigonometric function ***sin(x)***, you know, the wavy one. Now consider a related function, which henceforth I'll call ***cin(x)*** which has the defining property that ***cin(cin(cin(x))) = sin(x)***.

The Challenge:

Write a program or function which accepts an argument ***x*** in the range *[-Pi, Pi]* and outputs the corresponding value of ***cin(x)***. The faster and shorter the better but you should strive for *maximum accuracy* (at least 8-10 correct digits in the *whole* range, give or take a few *ulps*).

Once written, use it to tabulate ***cin(x)*** for *x = 0.0, 0.2, 0.4, ..., 1.0* and also to compute *cin(Pi/2), cin(-0.71), cin(2.019)*, and with the experience gained do likewise with another similar function *tin(x)* which has the property that *tin(tin(x)) = sin(x)*.

I'll give a short program for the **HP-71B** which achieves about 10 correct digits for any argument in *[-Pi, Pi]*.

Hint: You can check that you get adequately accurate values of ***cin(x)*** by simply computing ***cin(cin(cin(x)))*** and comparing the results with ***sin(x)***. Likewise with *tin(x)*.

Finally, the usual caveat:

- Please do **NOT** include ***CODE*** panels in your replies to this thread, as it makes it difficult for me to generate the online PDF document which will include the whole thread. I expect you'll kindly comply with this requirement but otherwise I'll remove from the final PDF document any replies featuring ***CODE*** panels. Thank you.

I'll post my original solutions in a week or so but meanwhile let's see what you can do.
That's all. Hope you'll enjoy it !

V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-21-2019, 01:21 PM **Post: #2**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Homage number must be divisible by 271 * 41 * 9 = 99999
Since 1e5 mod 99999 = 1, Homage number had another property:

top 5 digits + bottom 5 digits = 99999

Example, since 10234 + 89765 = 99999, smallest Homage number is 10234 89765

Top digit > 0, thus 9 cases to choose.
2nd digit cannot be the first, or 9 - first, thus 10 - 2 = 8 cases ...

Total Homage numbers = 9 * 8 * 6 * 4 * 2 = 3456

EMAIL PM FIND QUOTE REPORT

03-21-2019, 01:30 PM (This post was last modified: 03-21-2019 02:50 PM by Paul Dale.) **Post: #3**

Paul Dale
Senior Member

Posts: 1,662
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

I'm not sure why but tier 3 and tier 4 look like the easiest.

Spoilers towards the end.

Anyway, a 24 step program for tier 3 for the HP 25:

```
01 CLREG
02 1
03 x<>y
04 x=0?
05 GTO 24
06 2
07 /
08 INT
09 LASTx
10 FRAC
11 x=0?
```

```
12 GTO 17
13 Rv
14 x<>y
15 STO+ 0
16 GTO 19
17 Rv
18 x<>y
19 1
20 e^x
21 *
22 x<>y
23 GTO 04
24 RCL 0
```

For the Pi flavour, change steps 19 and 20 to: Pi and NOP.

The 1,000,000th term is 278,394,443.2 and the 3,141,593rd term is 1,601,007,657. Both in several seconds on a real 25.

The 1,234,567th term for Pi is 9,091,632,462 and the 2,718,282nd term is 30,446,503,22x (x being beyond the accuracy of the 25). Again, fairly quickly.

The key observation being that:

$$\sum_{i=0}^{n} e^i = \frac{e^{n+1}-1}{e-1} < e^{n+1}$$

which means that the position expressed in binary determines the powers used. Note that $2^1 + 2^3 + 2^4 = 26$ and $2^0 + 2^2 + 2^3 + 2^4 + 2^5 = 61$ and compare to the initial examples.

Larger bases also have this property. I didn't try to prove that *e* is the smallest base for which this holds true, which is good because the smallest base is, unsurprisingly, two.

Pauli

PM FIND QUOTE REPORT

03-22-2019, 09:04 AM **Post: #4**

J-F Garnier
Senior Member

Posts: 461
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

(I noticed that the SSMC24 was Valentin's post #314 at the time of writing :-)

Tier 5: Guru
In summary : Write a program or function cin(x) which has the defining property that cin(cin(cin(x))) = sin(x).
Once written, use it to tabulate cin(x) for x = 0.0, 0.2, 0.4, ..., 1.0

From the properties of sin we can restrict the search of cin(x) in the interval [0,pi/2].

The problem implicitly assumes that the cin(x) function is unique, otherwise it would make no sense to discuss the cin(x) values.
If we understand 'function' in the mathematic sense of an analytical, non-pathologic function, this may be true.

But if we understand 'function' in the computer science sense of a procedure that takes one argument and returns one result, there are very likely many cin 'functions' such as cin(cin(cin(x)))=sin(x)
One such solution for the HP71 is below:

```
10 ! SSMC24A
20 DEF FNC(X)
30 IF X=0 THEN Y=0 @ GOTO 70
40 X=ABS(X)
50 Y=LN(X)
60 IF Y<2 THEN Y=1+EXP(X) ELSE Y=SIN(LN(LN(X-1)-1))
70 FNC=Y
80 END DEF
85 !
90 FOR X=0 TO 1 STEP .1
100 PRINT FNC(FNC(FNC(X)));SIN(X)
```

```
110 NEXT X

>RUN
0 0
9.98334166508E-2 9.98334166468E-2
.198669330795 .198669330795
.295520206664 .295520206661
.389418342308 .389418342309
.479425538604 .479425538604
.564642473395 .564642473395
.644217687238 .644217687238
.717356090899 .7173560909
.783326909628 .783326909627
.841470984808 .841470984808
```

I don't take this solution too seriously :-)
Still searching for a better one...

J-F

EMAIL PM WWW FIND QUOTE REPORT

03-23-2019, 01:40 AM **Post: #5**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi J-F, long time no see !

> **J-F Garnier Wrote:** ⇒ (03-22-2019 09:04 AM)
> (I noticed that the SSMC24 was Valentin's post #314 at the time of writing :-)

Yes, that was intentional on my part. Notice also that Paul Dale's post just before yours is his post #**1444**, i.e. **38^2**, which has the maximum number of 4's (three) in which a square can end.

> **Quote:**
> The problem implicitly assumes that the cin(x) function is unique, otherwise it would make no sense to discuss the cin(x) values. If we understand 'function' in the mathematic sense of an analytical, non-pathologic function, this may be true.

Yes, that's the kind of *"function"* I intended.

> **Quote:**
> But if we understand 'function' in the computer science sense of a procedure that takes one argument and returns one result, there are very likely many cin 'functions' such as cin(cin(cin(x)))=sin(x)
> One such solution for the HP71 is below ...

Quite clever of you to find the trick, but certainly that's not what I intended as a solution. The trick will work for any function (not just *sin(x)*) and for any number of function compositions (not just *3*), but it's of no mathematical value or interest whatsoever.

Also, you overdid it a little, there's no need of *EXP's* or *LN's* to make it work, simply *adding* and later *subtracting* a suitable *constant* would do just as well (and faster). Finally, your line

```
30 IF X=0 THEN Y=0 @ GOTO 70
```

can be shortened to

```
30 IF X=0 THEN END
```

because if a multi-line user-defined function ends without assigning the return value then it simply returns 0 by default. And `IF X=0` can be replaced by `IF NOT X`

> **Quote:**
> I don't take this solution too seriously :-)

Indeed you shouldn't, neither do I ...

> **Quote:**
> Still searching for a better one...

Good luck with that, I'm sure you'll succeed and I'm eager to see what you come up with.

Very glad to see you post here, much appreciated. Thanks for your continued interest in my *S&SMC's* and have a nice weekend.

V.

*P.S.: What would it take to lure you into releasing a version of **Emu71** which would run on a **32-bit** or 64-bit Windows OS, or at least on Android ? Begging ? Bribing ? Taking some relative hostage ? Just plain ol' money ? You tell me, please, it's really affecting my productivity !!*

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-23-2019, 08:53 AM **Post: #6**

J-F Garnier
Senior Member

Posts: 461
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** (03-23-2019 01:40 AM)
> **Hi J-F**, long time no see !
> ...
> Very glad to see you post here, much appreciated. Thanks for your continued interest in my *S&SMC's* and have a nice weekend.

I'm still following your various challenges, even participated to the the last one.
And when I don't participate, it may just mean I have nothing interesting to contribute.

> **Quote:**
>
> > **Quote:**
> > Still searching for a better one...
>
> Good luck with that, I'm sure you'll succeed and I'm eager to see what you come up with.

I investigated a few ideas, without success up to now.

> **Quote:**
> *P.S.: What would it take to lure you into releasing a version of **Emu71** which would run on a **32-bit** or 64-bit Windows OS, or at least on Android ?*

It's a bit OT - well maybe not so much since the HP71 and Emu71 are your favourite tools for your challenges.
In short, the answer is that I don't have the competences to port my Emu71/Dos to Windows, MacOS, Linux, not to mention iOS, Android. However, Emu71/DOS is open source...
And you know for sure that there is already an excellent Emu71 running on Windows (32/64 bits), and also an HP71 emulator for Android.

J-F

EMAIL PM WWW FIND QUOTE REPORT

03-23-2019, 05:03 PM **Post: #7**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Perhaps cin(x) need a bit more restriction, not just cin(cin(cin(x))) = sin(x)

sin(cin(x)) = cin(sin(x))

cin(x) should be an odd function, shape like sin(x), with value between sin(x) and x

For domain [-pi/2, pi/2], **cin(x) ≈ (1 + x^2/9) sin(x)**

Example:

sin(½) = 0.4794255386
cin(cin(cin(½))) = 0.4791650974, relative error ~ -0.05%

sin(cin(½)) ≈ 0.473044
cin(sin(½)) ≈ 0.473050

EMAIL PM FIND QUOTE REPORT

03-23-2019, 10:49 PM **Post: #8**

rprosperi
Senior Member

Posts: 4,439
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** (03-23-2019 01:40 AM)
> *P.S.: What would it take to lure you into releasing a version of **Emu71** which would run on a **32-bit** or 64-bit Windows OS, or at least on Android ? Begging ? Bribing ? Taking some relative hostage ? Just plain ol' money ? You tell me, please, it's really affecting my productivity !!*

Valentin - Emu71/DOS runs great on Win7/8/10 (including x64) operating under DOSBOX, though of course not at the full speed it would run if running as native code.

So, if by productivity you mean having to change to a DOS machine to run Emu71, then the above would save a lot of time and hassle, however if raw Emu71 execution speed is the issue, this may not be a big increase in speed, if it increases at all.

As for using Christoph's EMU71/Win, you can remove the "run at actual speed" option and have much higher speed, but I don't know how it compares to native Emu71/DOS on your machine.

If you send me a sample program with relatively long run-time, I'd be happy to time it running on my PC under both Emu71/DOS/DOSBOX and Emu71/Win at it's max speed. This is an older PC with a XEON 3GHz CPU running Win7x64, so not the fastest, but it at least provides some relative performance numbers.

--Bob Prosperi

EMAIL PM FIND QUOTE REPORT

03-24-2019, 10:08 PM **Post: #9**

J-F Garnier
Senior Member

Posts: 461
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **J-F Garnier Wrote:** (03-22-2019 09:04 AM)
> ***Tier 5: Guru***
> *In summary : Write a program or function cin(x) which has the defining property that cin(cin(cin(x))) = sin(x).*
> *Once written, use it to tabulate cin(x) for x = 0.0, 0.2, 0.4, ..., 1.0*
> ...
> Still searching for a better one...

> **Albert Chan Wrote:** (03-23-2019 05:03 PM)
> cin(x) should be an odd function, shape like sin(x), with value between sin(x) and x
>
> For domain [-pi/2, pi/2], **cin(x) ≈ (1 + x^2/9) sin(x)**
>
> Example:
>
> sin(½) = 0.4794255386
> cin(cin(cin(½))) = 0.4791650974, relative error ~ -0.05%
>
> sin(cin(½)) ≈ 0.473044
> cin(sin(½)) ≈ 0.473050

Since cin(x) is an odd function, its Taylor expansion can be written as cin(x)=x+a*x^3+b*x^5+...
So my approach was to build the Taylor expansion of cin(cin(cin(x))) and identify it to the well known sinus expansion

sin(x)=x-x^3/3!+x^5/5!...
The x^3 term is easy to calculate and is just -(1/3!)/3=-1/18, in agreement with Albert's approximation (-1/3!+1/9).
The best I could calculate (by hand) was the x^5 term. Then I search for the x^7 term by try-and-error to minimize the error.
Here is my best approximation and results showing cin(cin(cin(x))) and sin(x):

```
10 ! SSMC24
30 A=-1/18 @ B=-7/1080 @ C=-.0015
40 DEF FNC(X)=X+A*X^3+B*X^5+C*X^7
50 FOR X=.1 TO 1 STEP .1
60 PRINT X;FNC(FNC(FNC(X)));SIN(X)
70 NEXT X

> RUN
.1 9.98334166706E-2 9.98334166468E-2
.2 .198669334313 .198669330795
.3 .295520279883 .295520206661
.4 .38941902196 .389418342309
.5 .479429531682 .479425538604
.6 .564659801456 .564642473395
.7 .644278060315 .644217687238
.8 .717533814261 .7173560909
.9 .783784102052 .783326909627
1 .842523170608 .841470984808
```

Another approach?

J-F

03-25-2019, 10:19 AM (This post was last modified: 03-27-2019 10:40 AM by Oulan.) **Post: #10**

Oulan
Member

Posts: 56
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Extending the approach of JF, you can use the following HP prime program

```
--------------------
#pragma mode( separator(., integer(h32) )
#cas
reduc(l,n):= BEGIN
LOCAL m;
m:=MAKELIST(0,z,0,n);
FOR Z FROM 0 TO n DO
m[n+1-Z]:=l[SIZE(l)-Z];
END;
RETURN m;
END;
fff():= BEGIN
LOCAL l,m,n,z,p,q,r;
LOCAL a15,a13,a11,a9,a7,a5,a3;
PURGE(a15,a13,a11,a9,a7,a5,a3);
z:={a15,0,a13,0,a11,0,a9,0,a7,0,a5,0,a3,0,1,0};
n:=15;
p:=poly2symb(z,x);
r:=poly2symb(z,y);
FOR Y FROM 1 TO 2 DO
q:=(p|x=r);
l:=symb2poly(q,y);
l:=reduc(l,n);
p:=poly2symb(l,x);
END;
m:=MAKELIST(0,Z,0,n);
FOR Z FROM 0 to n DO
IF (Z MOD 2) = 1 THEN
m[n+1-Z]:=((-1)^FLOOR(Z/2))/(Z!);
END;
END;
PRINT(l);
PRINT(m);
a3:=eval(solve(l[n-2]=m[n-2],a3)[1]);
// was a3:=solve(l[n-2]=m[n-2],a3);
```

```
a5:=eval(solve(l[n-4]=m[n-4],a5)[1]);
a7:=eval(solve(l[n-6]=m[n-6],a7)[1]);
a9:=eval(solve(l[n-8]=m[n-8],a9)[1]);
a11:=eval(solve(l[n-10]=m[n-10],a11)[1]);
a13:=eval(solve(l[n-12]=m[n-12],a13)[1]);
a15:=eval(solve(l[n-14]=m[n-14],a15)[1]);
RETURN [a3,a5,a7,a9,a11,a13,a15];
END;
#end
--------------------
```

to compute the coefficients of the taylor serie of CIN(x).
Use it in CAS mode : " k:=fff() 'enter' ", then " k 'enter' "

Be careful this program will take some times on a real Prime.

But the converging is very slow, coefficient up to x^15 follows but give only 10^-6 error near 1

-1/18 -7/1080 -643/408240 -13583/29393280 -29957/215550720 -24277937/648499737600
-6382646731/953294614272000

Btw can someone explain the warning displayed when solving for the coefficients ?

"Warning, ^ is ambiguous on non square matrices. Use .^ to apply ^ element by element."

I don't see any matrices solving here ... ok I saw the problem see listing. Sometimes list of list are not displayed with all brackets.

Anyway, there should be a better approach to solve this nice challenge

EDIT new version of program, avoid computing useless power

```
#pragma mode( separator(., integer(h32) )
#cas
mulT(a,b):= BEGIN
LOCAL n,p,j,k;
n:=SIZE(a);
p:=MAKELIST(0,j,1,n);
FOR j FROM 1 TO n DO
FOR k FROM 1 TO n+1-j DO
p[j+k-1]+=a[j]*b[k]; // test removed Thanks Albert
END;
END;
RETURN simplify(p);
END;

fff2(n):= BEGIN
LOCAL cin,ccin,si;
LOCAL p,q,xn,s;
LOCAL lvar,lexpr;
LOCAL a3,a5,a7,a9,a11,a13,a15,a17;
LOCAL a19,a21,a23,a25,a27,a29,a31,a33;
LOCAL a35,a37,a39,a41,a43,a45,a47,a49;
LOCAL vars;
PURGE(a3,a5,a7,a9,a11,a13,a15,a17);
PURGE(a19,a21,a23,a25,a27,a29,a31,a33);
PURGE(a35,a37,a39,a41,a43,a45,a47,a49);
PURGE(x,y);
vars:={1,a3,a5,a7,a9,a11,a13,a15,a17,a19,a21,a23,a25,a27,a29,a31,a33,a35,a37,a39,a41,a43,a45,a47,a49};
cin:=MAKELIST(IFTE(p MOD 2,vars[(p+1)/2],0),p,0,n);
ccin:=cin;
FOR q FROM 1 TO 2 DO
s:=MAKELIST(0,p,0,n);
s[1]:=ccin[1];
xn:=cin;
FOR p FROM 2 TO n+1 DO
s:=s+ccin[p]*xn;
IF p<=n THEN xn:=mulT(xn,cin);END;
END;
ccin:=s;
END;
si:=MAKELIST(IFTE(p MOD 2,((-1)^FLOOR(p/2))/(p!),0),p,0,n);
lvar:=MAKELIST(cin[p],p,4,n+1,2);
```

```
lexpr:=MAKELIST(ccin[p]=si[p],p,4,n+1,2);
s:=solve(lexpr,lvar);
RETURN s[1];
END;
#end
```

use with ff2(2*n+1) n from 2 to 24 (fff2(29) start to be long on a real G2), fff2(49) take few seconds on a virtual one.

PM FIND QUOTE REPORT

03-25-2019, 05:27 PM (This post was last modified: 03-25-2019 05:28 PM by Oulan.) **Post: #11**

Oulan
Member

Posts: 56
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Perhaps applying some series acceleration techniques such as Romberg-Richardson or Aitken could help ...
P.S. at least try to compute on an HP hand-held device those Taylor coefficient

PM FIND QUOTE REPORT

03-25-2019, 10:01 PM **Post: #12**

Gerson W. Barbosa
Senior Member

Posts: 1,361
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

hp 33s

```
C0001 LBL D
C0002 ENTER
C0003 ENTER
C0004 2
C0005 /
C0006 COS
C0007 SQRT
C0008 *
C0009 RTN

0.2 XEQ C XEQ C XEQ C -> 1.985105083E-01
0.5 XEQ C XEQ C XEQ C -> 4.7756191243E-1
1.0 XEQ C XEQ C XEQ C -> 8.4122242185E-1
```

At least this is a tiny program...

EMAIL PM FIND QUOTE REPORT

03-25-2019, 10:53 PM **Post: #13**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Calculate cin(x) taylor coefficients with XCas. Only need adjust vars for more terms.

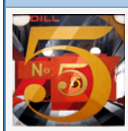
vars := [a3, a5, a7, a9, a11, a13, a15];

n := 2*len(vars) + 1;
y := x + sum(vars(k) * x^(2*k+1), k, 1, (n-1)/2);
cin(x0) := subst(y, x=x0);

solve(coeff(taylor(cin(taylor(cin(cin(x)), x, n)) - sin(x), x, n, polynom), x) = 0, vars)
→ [-1/18, -7/1080, -643/408240, -13583/29393280 ...]

EMAIL PM FIND QUOTE REPORT

03-28-2019, 01:38 AM **Post: #14**



Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, all:

First of all, thanks to all 5 of you who contributed to this thread so far, namely **Albert Chan**, **Paul Dale**, **J-F Garnier**, **Oulan** and **Gerson W. Barbosa**, your efforts and interest are much appreciated (and also thanks for not using *CODE* sections as I requested).

A week has elapsed and now I'll post my original solutions and comments to the different tiers discussed, one at a time, beginning with **Tier 1**:

Tier 1 - The Challenge:

We'll call ***Homage Number*** *to any 10-digit positive integers which are multiples of* ***271****, divisible by* ***41*** *and further their digits are all distinct. Write a program that takes no inputs but simply finds out and outputs just how many* ***Homage Numbers*** *there are*.

My original solution:

This tier was expressly created to be really **easy** so anyone interested could write code to solve it without much trouble. Matter of fact, as **Albert Chan** realized, it can be solved by hand with just a little thinking. The key is to realize that 41 and 271 are coprime, so every *Homage* number should be divisible by both and thus by their product, which is *41 * 271 = 11,111.*

Also, being a 10-digit number and having all its digits distinct means that its digits are *0, 1, 2, 3, ... , 9* in some order and thus their sum is *1 + 2 + 3 + ... + 9 = 45*, which is divisible by **9** so each *Homage* number has to be divisible by 9 too. As 9 is coprime to 41 and 271, each *Homage* number N must be divisible by their product, i.e., by *41 * 271 * 9 =* ***99,999****.*

Now let's split N into two 5-digit parts, **A**, **B**, like this: *N = 100,000*A + B*, which must be a multiple of 99,999, so subtracting *99,999*A* from it the resulting value: *100,000*A + B - 99,999*A =* ***A + B*** must be a multiple of 99,999 too and, as both A and B are 5-digit long, i.e., less than 100,000, that multiple must be 99,999 itself. Now, considering their individual digits we have:

A + B = abcde + uvxyz = **99999**

and thus all 5 pairs of digits must comply with *a + u = b + v = ... = e + z = 9*.

Now, there are **5!** permutations of the 5 pairs, so **120** permutations in all, but each of the 5 pairs has **2** possible orderings, say *(a,u)* and *(u,a)*, so $\mathbf{2^5 = 32}$ variations for each of the 120 permutations and thus there are *120 * 32 = 3,840 potential Homage* numbers in all.

However, only 9 out of 10 begin with a non-zero digit (numbers beginning with a 0 aren't 10-digits numbers) and so finally there are 3,840 * 9/10 = ***3,456 Homage Numbers***.

What if we can't or won't engage on such math reasoning ? Well, that's where our trusty **HP calc** will take away all the drudgery and work out the solution by itself, doing all the work for us in mere seconds and saving our neurons for better endeavours. In my case, this little *2-liner* for the **HP-71B** (fits in just 1 line too) will scan the whole range at steps of 99,999, increasing the count each time the corresponding number happens to have all its 10 digits different:

```
DESTROY ALL @ C=0 @ D=99999 @ FOR N=D*CEIL(10^9/D) TO 10^10-1 STEP D
C=C+NOT SPAN("0123456789",STR$(N)) @ NEXT N @ DISP C

>RUN
        3456
```

so there are *3,456 Homage Numbers* in all.

That's it. *Affordable*, as promised. In the next days I'll post my solutions for the subsequent tiers.

V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-28-2019, 12:10 PM **Post: #15**

Werner Posts: 552

Senior Member Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** ⇒ (03-28-2019 01:38 AM)
>
> **Hi, all**:
> A week has elapsed and now I'll post my original solutions and comments to the different tiers discussed, one at a time, beginning with **Tier 1**:
>
> [...]
>
> In the next days I'll post my solutions for the subsequent tiers.

Does that mean I'll have to wait 4 more days for the solution of tier 5? Dang.

Werner

EMAIL PM FIND QUOTE REPORT

03-28-2019, 01:51 PM (This post was last modified: 03-28-2019 02:07 PM by Albert Chan.) **Post: #16**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Homage number 99999 divisibility trick can be used for other numbers.
Example, to do *both* mod 7 and mod 13 at the same time, for X = 20 190 328

7 * 13 = 91
100 (mod 91) ≡ 9
1000 (mod 91) ≡ 90 ≡ -1

X (mod 91) ≡ 20 - 190 + 328 ≡ 158 ≡ 9 + 58 ≡ 67

X (mod 7) ≡ 67 - 63 ≡ 4
X (mod 13) ≡ 67 - 65 ≡ 2

EMAIL PM FIND QUOTE REPORT

03-29-2019, 02:55 AM **Post: #17**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, all:

Let's continue with my original solutions, today it's time for:

Tier 3 - The Challenge:

Those positive real numbers that are either powers or sums of distinct powers of an arbitrary real number ***P*** form an increasing sequence whose first term is **1** (i.e.: P^0). Write a program which accepts as input both ***P*** and an index ***k*** and returns the corresponding k^{th} term in the sequence.

Use your program to find the **1,000,000**th term and the **3,141,593**th term when ***P = e*** as well as the **1,234,567**th term and the **2,718,282**th term when ***P = Pi***.

My original solutions:

Though the concept used in this challenge will work for powers of any number ***P >= 2***, whether integer or real, I purposefully used a sequence of *real* numbers which were either powers or sums of distinct powers of ***e*** instead of powers of an *integer* to avoid giving as examples some integer sequences that people would immediately search for in *OEIS*.

For instance, my preliminary (not posted) examples where based on the integer sequences for ***P = 3*** and ***P = 61***, namely:

```
1 3 4 9 10 12 13 27 28 30 31 36 37 39 40 81 ...

1 61 62 3721 3722 3782 3783 226981 226982 227042 ...
```

but I decided to use instead ***P = e*** and ***P = Pi*** , which generate sequences of reals not present in *OEIS*. That said, the key fact is that the elements which are either powers or sum of distinct powers of a base ***P*** naturally *map* to the elements which are either powers or sum of distinct powers of base ***2***, which of course are all the integers *1, 2, 3...* , i.e. precisely the *indexes* for the elements in the base-***P*** sequence. Thus we only need to find the *base-**2*** expression for a given *index* and then interpret that base-***2*** expression as a number in base ***P***, which we then convert to the usual base 10. For example:

- to find the ***6th*** element in the sequence for ***P = 61***:

the index ***6*** in base 2 = 110_2 -> 110_{61} = $61^2 + 61^1$ = 3721 + 61 = **3782** in base 10

My original solution for the **HP-71B** is this *68-byte 1-liner*:

```
1  DEF FNE(N,K) @ M=0 @ P=1 @ REPEAT @ M=M+P*MOD(N,2) @ P=P*K @ N=N DIV 2 @ UNTIL NOT N @
FNE=M
```

and to compute the particular elements asked for in the challenge, simply:

```
>FNE(1000000,EXP(1))

        278394444.173 { = e^6 + e^9 + e^14 + e^16 + e^17 + e^18 + e^19 }

>FNE(3141593,EXP(1))

        1601007663.31

>FNE(1234567,PI)

         9091632437.43 { = Pi^0 + Pi^1 + Pi^2 + Pi^7 + Pi^9 + Pi^10 + Pi^12 + Pi^14 + Pi^15 + Pi^17 + Pi^20
}

>FNE(2718282,PI)

         30446503139.5
```

It's worth mentioning that for ***P = 8*** and ***P = 16*** there's an even *simpler* solution for the **HP-71B** right from the *command line*. For instance, to find the ***123th*** element in the sequence of powers or sum of distinct powers of ***8***, simply execute this from the command line:

```
>BVAL(BSTR$(123,2),8)

        299529
```

which of course agrees with the *1-liner*: `FNE(123,8) -> 299529.`

Also worth mentioning is the fact that my solution also works for ***P < 2***, even for ***P = 1***, ***P = 0*** and ***P < 0*** but then the resulting sequence is no longer in increasing order as is the case for ***P >= 2***. For instance:

```
P = 2       1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...           increasing, Ok

P = 1.9     1, 1.9, 2.9, 3.61, ..., 13.369, 13.0321, ... not increasing
P = Phi     1, 1.6180, 2.6180, 2.6180, 3.6180, ...       not increasing, repetitions
P = 1       1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, ...      ditto
P = 0       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, ...      ditto
P = -1      1, -1, 0, 1, 2, 0, 1, -1, 0, -2, -1, ...     ditto
P = -2      1, -2, -1, 4, 5, 2, 3, -8, -7, -10, ...      ditto
```

Last but not least, this is **my original solution** for the **HP-25**, a simple *24-step* affair:

```
01  STO 0     13   *
02  STO 1     14   RCL 1
03  STO/ 1    15   *
04  CLX       16   +
05  X<>Y      17   RCL 0
```

```
06  2        18  STO* 1
07  /        19  Rv
08  INT      20  X<>Y
09  X<>Y     21  X#0
10  LASTX    22  GTO 06
11  FRAC     23  X<>Y
12  2        24  GTO 00
```

For instance:

```
FIX 0
26, ENTER, 3, R/S -> 111
61, ENTER, 3, R/S -> 361
1000000, ENTER, 3, R/S -> 1726672221
```

So much for **Tier 3**, thanks a lot to **Paul Dale** for his interest in this particular tier and for taking the time to create a nice *24-step* solution for the **HP-25** as well. In the next days I'll post my solutions for the subsequent tiers.

V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-31-2019, 03:51 PM **Post: #18**

Juan14
Junior Member

Posts: 36
Joined: Jan 2014

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

For the last challenge.
Let cin(x) = sin(x+A), we have:
sin(sin(sin(x+A)+A)+A) = sin(x) or
sin(sin(x+A)+A)+A = x
For a given value of x, we can solve the last equation for A (A is a function of x).

Here is the program for the hp 50g:

```
«
→ x
«
'SIN(SIN(x+A)+A)+A-x'
'A' 1 ROOT x + SIN →NUM
»
»
```

x_____cin(x)
0______0.
0.2___0.19954743606
0.4___0.39617257453
0.6___0.586447829132
0.8___0.761 006258889
1______0.906981195071
π/2_____.835085096711
-0.71__-0.684625012855
2.019__4.81961624069E-3

There are so many ways to define the function cin(x) in a similar way, that's why I didn't post my solution before, but here it is anyway :-)

EMAIL PM FIND QUOTE REPORT

03-31-2019, 04:26 PM (This post was last modified: 03-31-2019 07:10 PM by Albert Chan.) **Post: #19**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Juan14 Wrote: (03-31-2019 03:51 PM)

For the last challenge.
Let cin(x) = sin(x+A), we have:
sin(sin(sin(x+A)+A)+A) = sin(x) or
sin(sin(x+A)+A)+A = x
For a given value of x, we can solve the last equation for A (A is a function of x).

Nice try, but identity cin(cin(cin(x))) = sin(x) does not hold

Example, with above cin(x) definition, and x = 1.0

sin(1) ≈ 0.841471
cin(cin(cin(1))) ≈ cin(cin(0.906981)) ≈ cin(0.844196) ≈ 0.796542

Even worse if x=Pi/2

sin(Pi/2) = 1
cin(cin(cin(Pi/2))) ≈ cin(cin(0.835085)) ≈ cin(0.789340) ≈ 0.752222

EMAIL PM FIND QUOTE REPORT

03-31-2019, 11:11 PM **Post: #20**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Valentin Albillo Wrote: (03-21-2019 03:08 AM)

Tier 4: Advanced

*[MRM: **HP-11C** and up]*

Consider the ***n***-point dataset (x_i , y_i) where x_i = *1, 2, 3, 4, 5, 6, ..., **n*** (the natural numbers) and y_i = *2, 3, 5, 7, 11, 13, ..., p_n* (the *prime* numbers), and the *(n-1)*st *degree polinomial fit* to this dataset of the form:

$$P(x) = a_0 + a_1 (x-1) + a_2 (x-1)(x-2) + \dots + a_{n-1} (x-1)(x-2)(x-3) \dots (x-(n-1))$$

Getting sum of above coefficients can be done by doing forward difference of the primes:

2 3 5 7 11 13 17 19 23 29 ... ; primes
1 2 2 4 2 4 2 4 6 ... ; Δ
1 0 2 -2 2 -2 2 2 ... ; Δ^2
-1 2 -4 4 -4 4 0 ... ; Δ^3
3 -6 8 -8 8 -4 ... ; Δ^4
...

$a_k = \Delta^k(0) / k!$

$\Sigma(a_k$, k = 0 to Inf) = 2/0! + 1/1! + 1/2! - 1/3! + 3/4! + ...

Sum converge very fast:
2.0
3.0
3.5
3.33333 333333
3.45833 333333
3.38333 333333
3.41527 777778
3.40476 190476
3.40761 408730
3.40696 097884
3.40708 691578
3.40706 684905 ; 6 digits accuracy with 12 primes
3.40706 938140
3.40706 915834
3.40706 916344
3.40706 916625
3.40706 916552
3.40706 916564
3.40706 916563 ; 12 digits accuracy with 19 primes

04-01-2019, 01:49 AM **Post: #21**

Juan14
Junior Member

Posts: 36
Joined: Jan 2014

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

You are right Albert and I can't find a way around.

04-01-2019, 04:25 PM (This post was last modified: 04-03-2019 07:01 PM by Albert Chan.) **Post: #22**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Just figured out how to improve cin(x) accuracy for large x :)

cin(x) = arcsin(cin(sin(x))) = nest(arcsin, cin(nest(sin, x, n)), n)

Pick enough nested sin's so cin argument is small, say below 0.1 radian

```
cin[x0_] := Block[ {n=0, x=x0+0.0},
    While[Abs[x] ≥ 0.1, x = Sin[x]; n++];
    Nest[ArcSin, x - (1/18) x^3 - (7/1080) x^5 - (51/32285) x^7, n]
]
```

Above cin(x) setup give about 12 digits accuracy:

x	**cin(x)**	**cin(cin(cin(x))) - sin(x)**
0.0	0.0	+0.0
0.2	0.199553461081	-1.9e-16
0.4	0.396375366278	+1.8e-14
0.6	0.587446695546	-1.1e-16
0.8	0.769025184826	-9.1e-14
1.0	0.935745970819	+1.4e-13
Pi/2.	1.210368344457	+2.6e-13
-0.71	-0.688778525307	-1.6e-13
2.019	1.026923318694	+6.4e-13

Edit: changed x^7 coefficient from -0.00158 to -51/32285 to get better accuracy

04-01-2019, 05:42 PM (This post was last modified: 04-01-2019 07:29 PM by J-F Garnier.) **Post: #23**

J-F Garnier
Senior Member

Posts: 461
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Albert Chan Wrote:** ⇒ (04-01-2019 04:25 PM)
>
> Just figured out how to improve cin(x) accuracy for large x :)
>
> cin(x) = arcsin(cin(sin(x))) = nest(arcsin, cin(nest(sin, x, n)), n)
>
> Pick enough nested sin's so cin argument is small, say below 0.1 radian
>
> ```
> cin[x0_] := Block[ {n=0, x=Evaluate[x0+0.0]},
>     While[Abs[x] ≥ 0.1, x = Sin[x]; n++];
>     Nest[ArcSin, x - (1/18) x^3 - (7/1080) x^5 - 0.00158 x^7, n]
> ]
> ...
> ```

Excellent !
Here is the HP71 version and results, after decipher of your code (not familiar with that language...):

```
10 ! SSMC24
20 A=-1/18 @ B=-7/1080 @ C=-.00158
30 DEF FNC(X)
40 N=0
50 X=SIN(X) @ N=N+1 @ IF ABS(X)>=.1 THEN 50
```

```
60 ! X=X+A*X^3+B*X^5+C*X^7
61 X=C*X^7+B*X^5+A*X^3+X ! better
70 FOR I=1 TO N @ X=ASIN(X) @ NEXT I
80 FNC=X
90 END DEF
100 !
110 FOR X=.2 TO 1 STEP .2
120 Y=FNC(FNC(FNC(X)))
130 PRINT X;Y;SIN(X);Y-SIN(X)
140 NEXT X

>RUN
.2 .198669330795 .198669330795 0
.4 .389418342314 .389418342309 .000000000005
.6 .564642473542 .564642473395 .000000000147
.8 .717356091570 .717356090900 .000000000670
1. .841470984040 .841470984808 -.000000000768

>FNC(PI/2);FNC(FNC(FNC(PI/2)))
1.2103683495 .999999998579

>FNC(-0.71)
-.688778525229

>FNC(2.019)
1.02692332142
```

J-F
[Edited: reversed the order of the polynom term evaluation, for slightly better accuracy]

EMAIL PM WWW FIND QUOTE REPORT

04-02-2019, 11:43 PM **Post: #24**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, all:

Continuing with my original solutions, today it's time for:

Tier 4 - The Challenge:

*Consider the **n**-point dataset (1, 2), (2, 3), (3, 5), (4, 7), (5, 11), (6, 13), ..., (n, p_n) (the prime numbers), and the $(n-1)^{st}$ degree polinomial fit to this dataset of the form:*

$$P(x) = a_0 + a_1 (x-1) + a_2 (x-1)(x-2) + \ldots + a_{n-1} (x-1)(x-2)(x-3) \ldots (x-(n-1))$$

*Write a program that takes no inputs but computes and outputs the limit of the sum of the coefficients a_0, a_1, ..., a_{n-1} when **n** tends to infinity.*

My original solution:

My original solution for the **HP-71B** is this *4-liner* (168 bytes):

```
1  DESTROY ALL @ OPTION BASE 0 @ REPEAT @ N=N+1 @ DIM C(N) @ T=S
2  FOR I=1 TO N @ C(I)=FPRIM(C(I-1)+1) @ NEXT I @ S=0
3  FOR I=1 TO N-1 @ FOR J=N TO I+1 STEP -1 @ C(J)=C(J)-C(J-1) @ NEXT J @ NEXT I
4  FOR I=1 TO N @ S=S+C(I)/FACT(I-1) @ NEXT I @ UNTIL S=T @ DISP N;S

>RUN
        20    3.40706916561     { it converged to the limit after fitting the first 20 primes:
2, 3, 5, ..., 71) }
```

Notes:

- *Line **1*** initializes and begins the loop to compute the sum of the first *n* coefficients
- *Line **2*** fills an array with the first *n* primes

- *Line 3* computes the forward differences in-place (replacing the primes)
- *Line 4* computes the sum of the coefficients (differences / factorials) and loops back until it agrees with the previous sum, then outputs it

That's all for **Tier 4**, thanks a lot to **Albert Chan** for his interest in this particular tier and congratulations for providing a correct solution and some explanation but please, **Albert**, next time **do* provide actual code *for an HP calculator* of your choice, so that people can try your solution for themselves.

In the next days I'll post my solutions for the remaining tiers.

V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

04-03-2019, 03:05 AM **Post: #25**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** (04-02-2019 11:43 PM)
>
> >RUN
> 20 **3.40706916561** *{ it converged to the limit after fitting the first 20 primes: 2, 3, 5, ..., 71) }*

I think you meant sum converged using 19 primes (20 primes to confirm 12-digits convergence)

sum using 19 primes = **414453 270752** 384363 / 19! ≈ 3.40706 916563
sum using 20 primes = **414453 270752 5**80132 / 19! ≈ 3.40706 916563

Also, forward difference tables may be built incrementally.

C(1) = p1
C(2) = p2 - p1
C(3) = p3 - 2 p2 + p1,
C(4) = p4 - 3 p3 + 3 p2 - p1,
C(5) = p5 - 4 p4 + 6 p3 - 4 p2 + p1,
...

Above can be simplified without a prime table:

C(1) = p1
C(2) = p2 - C(1)
C(3) = p3 - C(1) - 2 C(2)
C(4) = p4 - C(1) - 3 C(2) - 3 C(3)
C(5) = p5 - C(1) - 4 C(2) - 6 C(3) - 4 C(4)
...

EMAIL PM FIND QUOTE REPORT

04-03-2019, 03:57 AM (This post was last modified: 04-27-2019 06:02 PM by Albert Chan.) **Post: #26**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

I only have a HP-12C, which is not powerful enough to make primes, build delta tables ...

XCas code:

```
terms(n) := {
local c, s, p, j, k;
c := flatten(matrix(n,0)); s := 0; p := 0;
for(j:=0; j<n; j++) {
    p := nextprime(p);
    c[j] := p;
    for(k:=0; k<j; k++) c[j] := c[j] - comb(j,k) * c[k];
```

```
    s += c[j] / float(j!);
    print(p, s);
    }
}
```

terms(20) →

```
02 2.0
03 3.0
05 3.5
07 3.33333333333
11 3.45833333333
13 3.38333333333
17 3.41527777778
19 3.40476190476
23 3.4076140873
29 3.40696097884
31 3.40708691578
37 3.40706684905
41 3.4070693814
43 3.40706915834
47 3.40706916344
53 3.40706916625
59 3.40706916552
61 3.40706916564
67 3.40706916563
71 3.40706916563
```

Edit: replaced Python code to XCas, so HP prime user can try out.

04-05-2019, 02:58 AM **Post: #27**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, all:

At long last, today it's time for my final *original solution*, namely:

Tier 5 - The Challenge:

Consider the function ***cin(x)*** which has the defining property that ***cin(cin(cin(x))) = sin(x)***.

Write a program or function which accepts an argument ***x*** in the range *[-Pi, Pi]* and outputs the corresponding value of ***cin(x)*** correct to at least 8-10 digits in the *whole* range. Use it to tabulate ***cin(x)*** for *x = 0.0, 0.2, 0.4, ..., 1.0* and also to compute *cin(Pi/2), cin(-0.71), cin(2.019)*.

My original solution:

My original solution for the **HP-71B** is the following *user-defined function* (plus initialization code):

```
DESTROY ALL @ OPTION BASE 1 @ DIM C(7) @ READ C
DATA 1,-1/18,-7/1080,-643/408240,-13583/29393280,-29957/215550720,-24277937/648499737600

DEF FNC(X) @ L=0 @ M=1/3 @ REPEAT @ X=SIN(X) @ L=L+1 @ UNTIL ABS(X)<M
S=0 @ FOR Z=1 TO 7 @ S=S+C(Z)*X^(2*Z-1) @ NEXT Z
FOR Z=1 TO L @ S=ASIN(S) @ NEXT Z @ FNC=S @ END DEF
```

Instead of tabulating it for *0.0, 0.2, ..., 1.0* as I originally asked, let's better tabulate it for *x* from *0* to *Pi/2* in steps of *Pi/10*:

```
FOR X=0 TO PI/2 STEP PI/10
Y=FNC(FNC(FNC(X))) @ DISP X;FNC(X);Y;SIN(X);Y-SIN(X) @ NEXT X

>FIX 10
>RUN
```

x	cin(x)	cin(cin(cin(x)))	sin(x)	Error
0.0000000000	*0.0000000000*	0.0000000000	0.0000000000	0
0.3141592654	*0.3124163699*	0.3090169944	0.3090169944	-1.0E-12
0.6283185307	*0.6138343796*	0.5877852523	0.5877852523	2.2E-11
0.9424777961	*0.8897456012*	0.8090169944	0.8090169944	4.1E-11
1.2566370614	*1.1122980783*	0.9510565164	0.9510565163	1.0E-10
1.5707963268	*1.2103683445*	1.0000000000	1.0000000000	1.0E-11

So we've got 10 correct decimals or better, as the error in *cin(x)* is even smaller than the error in *cin(cin(cin(x)))-sin(x)* which doesn't exceed 10^{-10}. As for the discrete values asked in the challenge:

```
>FIX 10 @ FNC(PI/2); FNC(-0.71); FNC(2.019)

        1.2103683445   -0.6887785253   1.0269233188
```

Notes:

- ***Line 4*** evaluates the formal series in a simple loop but that's not optimal. I know of several better ways to evaluate the series but I don't want to digress from the main subject, which is the computation of ***cin(x)***.

- **Albert Chan** found the correct *conjugation (sin/arcsin)* procedure to increase accuracy and *almost* duplicated my original solution but there's an important difference which affects both accuracy and running time. He used up to the x^7 term in his formal series expansion:

  $x - (1/18)\, x^3 - (7/1080)\, x^5 - 0.00158\, x^7$

  and then iterated the sine of the argument till it got **< 0.1**, while my original solution uses up to the x^{13} term:

  $x - 1/18\, x^3 - 7/1080\, x^5 - 643/408240\, x^7 - 13583/29393280\, x^9 - 29957/215550720\, x^{11} - 24277937/648499737600\, x^{13}$

  and iterates until the *sin* gets **< 1/3**. This way significantly *fewer sin/arcsin* iterations are needed and the computation is both more accurate and faster. For instance, to see how many iterations my function performs when computing ***cin(Pi/2)*** just execute this:

  >FNC(PI/2);L
  1.2103683445 **24**

  thus **24** *sin/arcsin* were necessary for this argument while in **J-F Garnier**'s *HP-71B* version of Albert Chan's code several hundred sines/arcsines are necessary to bring this argument below 0.1, which explains why it takes much longer and worse, several decimal places are *lost* in the process.

- My solution will also work for ***tin(x)***, defined as ***tin(tin(x)) = sin(x)***, by simply replacing the coefficients in the `DATA` statement at line 2 by those of its own formal series, namely:

  $x - x^3/12 - x^5/160 - 53/40320\, x^7 - 23/71680\, x^9 - 92713/1277337600\, x^{11} + \ldots$

  and of course it will also work for any other such functions as well.

- The coefficients of the formal series for ***cin(x)*** and ***tin(x)*** can be obtained in a number of ways (even manually for the first 4 or so !), most easily by using some *CAS* which can deal with formal series (even a version of *Newton's method* for solving *f(x) = 0* can be put to the task), but it's important to be aware that both formal series *do **not** converge*.

  In fact, their *radius of convergence* is **0** and thus they behave like *asymptotic* series, so you can't get arbitrarily accurate results by taking more and more terms, you must instead truncate the series after a certain number of terms to get the most accurate results. Using further terms only *worsens* the accuracy.

- Although at first sight the coefficients of the formal series for ***cin(x)*** and ***tin(x)*** seem to (slowly) get smaller and smaller, matter of fact they tend to grow ever bigger after a while, tending to infinity. For instance, for ***tin(x)*** we find that the smallest coefficient in absolute value is:

  $Coeff_{37} = -0.00000000594338574503$

  but afterwards we have, e.g.:

  $Coeff_{101} = 0.0833756228055$

  $Coeff_{151} = 388536047335.239$

$Coeff_{201}$ = 6555423874651256623811186991.51

$Coeff_{251}$ = -35365220492708296140377087748804440170254492009.57

That's all for **Tier 5**, I could say a *whole* lot more about this topic and post additional code and results but this post is long enough as it is so I'll stop right now.

Thank you very much to **Albert Chan, J-F Garnier**, **Oulan** and **Gerson W. Barbosa** for your valuable contributions and to **Werner** for your interest, I hope you enjoyed it all !

V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

04-05-2019, 08:40 PM (This post was last modified: 04-07-2019 05:12 PM by Albert Chan.) **Post: #28**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Below Lua code scale cin argument to [sin(0.5), 0.5], do cin, then undo asin/sin's

```
local sin, asin = math.sin, math.asin

function cin(x)
    local y, n = x*x, 0
    while y > 0.25 do x=sin(x); y=x*x; n=n+1 end
    if y < 0.0324 then   -- |x| < 0.18
        local z = y*(0.00013898 + y*0.00003744) + 13583/29393280
        x = x - x*y*(1/18 + y*(7/1080 + y*(643/408240 + y*z)))
        return n==0 and x or asin(x)
    end
    while y < 0.229848847 do x=asin(x); y=x*x; n=n-1 end

    y = y - 0.2399        -- |x| = [sin(0.5), 0.5]
    y = 0.013724194890539722 + y*(
          0.058965322546572385 + y*(
          0.007795773378183463 + y*(
          0.002109528417736682 + y*(
          0.000663984666232017 + y*(
          0.000199482968029459 )))))

    x = x - x*y             -- x = cin(x)
    for i=1,n do x = asin(x) end
    for i=1,-n do x = sin(x) end
    return x
end
```

Result *very* accurate. Example:

```
x = 2.019
cin(x) = 1.02692 331869 35764
cin(cin(x)) = 0.956628 929996 1186
cin(cin(cin(x))) = 0.90122 698939 98129
math.sin(x)      = 0.90122 698939 98126
```

EMAIL PM FIND QUOTE REPORT

04-07-2019, 05:58 PM **Post: #29**

John Keith
Senior Member

Posts: 615
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Though I did not participate in this challenge, I have taken the liberty of adapting Valantin's Albert's programs into RPL with a twist- unlimited precision.

This program does not run until convergence but a fixed number of iterations, which is the number n that is input into the program. The program requires the external libraries ListExt, GoferLists, and Long Float.

```
%%HP: T(3)A(R)F(.);
@ Generate list of primes:
\<< \-> n
\<< 2 2 n
START DUP NEXTPRIME
NEXT n \->LIST
@ Inverse binomial transform of above list:
DUP HEAD SWAP 2 n
START \GDLIST DUP HEAD SWAP
NEXT DROP n \->LIST
@ List of factorials 0 through n:
1 n 1 - LSEQ
\<< *
\>> Scanl1 +
@ Divide to form list of ratios:
/
@ Accumulate above list:
\<< + EVAL
\>> Scanl1
@ Convert to list of LongFloats:
1.
\<< \->FNUM
\>> DOLIST
\>>
\>>
```

To begin, store a number into the variable DIGITS which sets the precision that LongFloat uses. In this example, I used 50. for DIGITS and 60 for the number of iterations.

I then used the following simple program to turn the resulting list into a string suitable for display or printing:

```
\<< \->STR 3. OVER SIZE 2. - SUB " " 13. CHR 10. CHR + SREPL DROP
\>>
```

The result:

```
2
3
35000000000000000000000000000000000000000000000000.E-49
33333333333333333333333333333333333333333333333333.E-49
34583333333333333333333333333333333333333333333333.E-49
33833333333333333333333333333333333333333333333333.E-49
34152777777777777777777777777777777777777777777778.E-49
34047619047619047619047619047619047619047619047619.E-49
34076140873015873015873015873015873015873015873016.E-49
34069609788359788359788359788359788359788359788360.E-49
34070869157848324514991181657848324514991181657848.E-49
34070668490460157126823793490460157126823793490460.E-49
34070693813966383410827855272299716744161188605633.E-49
34070691583365194476305587416698527809638920750032.E-49
34070691634410012386202862393338583814774290964767.E-49
34070691662452161790786129410468034806659145283484.E-49
34070691655244232025812713643401474089304777135465.E-49
34070691656406347881896434650869758246042092914175.E-49
34070691656257873262373750840742575708166882908384.E-49
34070691656273966717789714824786163263074495403684.E-49
34070691656272442599684037804120145048719207427525.E-49
34070691656272570305882488238644909673728392032261.E-49
34070691656272560845399289953795750049707556470279.E-49
34070691656272561452781304231311072994035926413957.E-49
34070691656272561421162961843454416475655340721354.E-49
34070691656272561422168859510781044267796095465102.E-49
34070691656272561422203623227227792237954574766870.E-49
34070691656272561422193499936605021028268165932935.E-49
34070691656272561422194680710735493212204028544306.E-49
34070691656272561422194575066153490058695686270606.E-49
34070691656272561422194583144322191113049301542936.E-49
34070691656272561422194582596611005303547258196512.E-49
```

34070691656272561422194582630026111767164206858021.E-49
34070691656272561422194582628184366702426127844481.E-49
34070691656272561422194582628275666348866780110705.E-49
34070691656272561422194582628271661692858309901367.E-49
34070691656272561422194582628271810600936280034971.E-49
34070691656272561422194582628271806504336081216950.E-49
34070691656272561422194582628271806529151697629234.E-49
34070691656272561422194582628271806536170465629760.E-49
34070691656272561422194582628271806535499300745912.E-49
34070691656272561422194582628271806535542548688840.E-49
34070691656272561422194582628271806535540241528738.E-49
34070691656272561422194582628271806535540348557090.E-49
34070691656272561422194582628271806535540344239572.E-49
34070691656272561422194582628271806535540344383289.E-49
34070691656272561422194582628271806535540344380187.E-49
34070691656272561422194582628271806535540344380140.E-49
34070691656272561422194582628271806535540344380151.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380149.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380149.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380151.E-49
34070691656272561422194582628271806535540344380150.E-49
34070691656272561422194582628271806535540344380151.E-49

It can be observed that:

-- LongFloat numbers are not very user-friendly.

-- There is noise in the last digit, so really 49-digit accuracy in this case.

-- Rate of converge increases, only about 56 iterations required to confirm 49 digits.

04-08-2019, 04:36 PM (This post was last modified: 04-10-2019 03:41 AM by Albert Chan.) **Post: #30**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

I posted cin(x) puzzle to the Lua mailing list, and got an elegant solution from Egor Skriptunoff.
Taylor coefficients built on the fly, without any need for CAS.
http://lua-users.org/lists/lua-l/2019-04/msg00063.html

Below code modified a bit for speed, accuracy, and extended cin(x) for tin(x):

> **Quote:**
```
local sin, asin = math.sin, math.asin

local function g(k, m, c, a)     -- assume c[0] = 1, m = [0,999]
    if k < 2 then return c[m] end
    local i = 1000 * k + m
    local r = a[i]
    if r then return r end
    r = g(k-1, m, c, a) + c[m] -- case for j=0 and j=m
    for j = 1, m-1 do
        r = r + c[j] * g(k-1, m-j, c, a)
    end
    a[i] = r
    return r
end

local function f(d, c, a)
    local r = 0
    for j = 1, #c do
        r = r + d[j] * g(2*j+1, #c+1-j, c, a)
    end
```

```
    return r
end

function maclaurin_of_cin()
    local c, c2, s = {}, {}, 1
    return function(k)
        for n = #c + 1, k do
            s = -(2*n)*(2*n+1)*s
            local a = {}
            local r, r2 = f(c, c, a), f(c2, c, a)
            local t = (1/s-r-r2)/3
            c[n], c2[n] = t, r + 2*t
        end
        return c[k]
    end
end

function maclaurin_of_tin()
    local c, s = {}, 1
    return function(k)
        for n = #c + 1, k do
            s = -(2*n)*(2*n+1)*s
            c[n] = (1/s - f(c, c, {})) / 2
        end
        return c[k]
    end
end

function egor(x)
    if x*x > 0.25 then return asin(egor(sin(x))) end
    local r, p, s, n, R = 0, x, x*x, 0
    repeat
        R, n, p = r, n+1, p*s
        r = r + maclaurin_coefs(n) * p
    until r == R
    return r + x
end
```

```
lua> maclaurin_coefs = maclaurin_of_tin()
lua> for i=50,125,25 do     -- match post #28 Coefs
:       print(2*i+1, maclaurin_coefs(i))
:       end
101  0.08337562280550574
151  388536047335.2163
201  6.555423874650777e+027
251  -3.536522049267692e+046

lua> function nest(f,x,n) for i=1,n do x=f(x);print(i, x) end end
lua> nest(egor, 2.019, 2) -- egor = tin
1       0.9894569770589354
2       0.9012269893998129

lua> maclaurin_coefs = maclaurin_of_cin()
lua> nest(egor, 2.019, 3) -- egor = cin
1       1.0269233186935764
2       0.9566289299961186
3       0.9012269893998129

lua> math.sin(2.019)
0.9012269893998126
```

EMAIL PM FIND QUOTE REPORT

04-09-2019, 07:53 PM **Post: #31**

Gerson W. Barbosa
Senior Member

Posts: 1,361
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

John Keith Wrote: ⇒ (04-07-2019 05:58 PM)

It can be observed that:

-- LongFloat numbers are not very user-friendly.

They needn't be so.

34070691656272561422194582628271806535540344380151.E-49

\<< ZZ\<-\->F -51 FC? { "." } { "," } IFTE SWAP ROT \->STR DUP SIZE ROT + OVER 1 ROT SUB ROT + " " ROT + 1 ROT REPL
\>>

EVAL

-->

3.4070691656272561422194582628271806535540344380151

EE7Dh
100 bytes,

which can be optimized, of course.

EMAIL PM FIND QUOTE REPORT

04-10-2019, 12:37 AM **Post: #32**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

.
Hi again, all

> **Albert Chan Wrote:** (04-05-2019 08:40 PM)
> Below Lua code scale cin argument to [sin(0.5), 0.5], do cin, then undo asin/sin's [...] **Result *very* accurate**.
> Example:
> x = 2.019
> cin(x) = 1.02692 331869 35764
> cin(cin(x)) = 0.956628 929996 1186
> cin(cin(cin(x))) = 0.90122 698939 98129
> math.sin(x)   = 0.90122 698939 98126

Indeed, *impressive* accuracy ! Thanks a lot for your Lua code, **Albert Chan**, I hope you'll adapt it to some HP calc's native programming language when you eventually get your hands on one (apart from the **HP-12C**, that is).

> **John Keith Wrote:** (04-07-2019 05:58 PM)
> Though I did not participate in this challenge, I have taken the liberty of adapting Valantin's Albert's programs into RPL with a twist- unlimited precision.[...] The result:
> [...]
> 34070691656272561422194582628271806535540344380151.E-49
> [...]
> -- Rate of converge increases, only about 56 iterations required to confirm 49 digits.

Yes, it does converge very fast and I *love* multiprecision computations and results. In fact, I don't understand why *HP* didn't ever include it in some of its advanced models right from the box (at least *double precision* as in some *SHARP* models which would do 20 digits without batting an eyelid.)

Thanks a lot for your interest and your *RPL high-precision* results, much appreciated.

> **Albert Chan Wrote:** (04-08-2019 04:36 PM)
> I posted cin(x) puzzle to the Lua mailing list, and got an elegant solution from Egor Skriptunoff. Taylor coefficients built on the fly, without any need for CAS.
> http://lua-users.org/lists/lua-l/2019-04/msg00063.html

[...]
Below code modified a bit for speed, accuracy, and extended **cin(x)** for **tin(x)**:
[...]
lua> function nest(f,x,n) for i=1,n do x=f(x);print(i, x) end end
lua> nest(egor, 2.019, 2) -- egor = **tin**
1 0.9894569770589354
2 0.9012269893998129

lua> maclaurin_coefs = maclaurin_of_cin()
lua> nest(egor, 2.019, 3) -- egor = **cin**
1 1.0269233186935764
2 0.9566289299961186
3 0.9012269893998129

lua> math.**sin**(2.019)
0.9012269893998126

As I said before, *truly excellent* accuracy. Also thank you very much for posting my challenge to the **Lua** forums, for giving me credit for it, and for your outstandingly clear code which also includes an implementation and high-precision results fot the **tin(x)** function I mentioned in the challenge. Again, really appreciated.

> **Gerson W. Barbosa Wrote:** (04-09-2019 07:53 PM)
>
> > **John Keith Wrote:** (04-07-2019 05:58 PM)
> >
> > It can be observed that: [...] LongFloat numbers are not very user-friendly. :)
>
> They needn't be so.
>
> 340706916562725614221945826282718065355403443801 51.E-49
> [...]
> 3.40706916562725614221945826282718065355403443801 51

Very good effort to increase usability. As you know RPL is not my thing but I can appreciate your ingenuity. Thanks, **Gerson**.

Best regards to all of you.
V.
.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

04-10-2019, 05:11 PM (This post was last modified: 04-10-2019 05:11 PM by John Keith.) **Post: #33**

John Keith
Senior Member

Posts: 615
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** (04-10-2019 12:37 AM)
>
> Yes, it does converge very fast and I *love* multiprecision computations and results. In fact, I don't understand why *HP* didn't ever include it in some of its advanced models right from the box (at least *double precision* as in some *SHARP* models which would do 20 digits without batting an eyelid.)
>
> Thanks a lot for your interest and your *RPL high-precision* results, much appreciated.

Thanks for your kind words, Valentin. The HP 49 and 50 do have exact integers whose size is limited only by memory. Though LongFloat is an external library and is a bit rough around the edges, its precision can be set up to 9999 digits. At that point, I think formatting becomes moot. :)

EMAIL PM FIND QUOTE REPORT

04-13-2019, 09:01 PM **Post: #34**

Bernd Grubert
Member

Posts: 91
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hello Valentin,
I don't understand the term composite in the context of Tier 2. I first thought, that the result of S_B must have at least 2 digits, but that can't be the point. Please explain what's meant by composite.

Best regards
Bernd

EMAIL PM FIND QUOTE REPORT

04-13-2019, 11:35 PM (This post was last modified: 04-14-2019 09:53 PM by Albert Chan.) **Post: #35**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

I recently created nextprime.lua, which is needed for solving Tier 2 puzzle.
My Lua code available in https://github.com/achan001/PrimePi

Quote:

```
p = require 'nextprime'

function sb(base, n)
    local t, d = 0
    while n > 0 do
        d = n % base; t = t + d; n = (n-d)/base
    end
    return t
end

function sb_find(base, n)
    if not n then n=1 end
    return function()
        repeat n = p.nextprime(n) until not p.isprime(sb(base, n))
        return n
    end
end
```

```
lua> function loop(n,f) for i=1,n do io.write(f(),' ') end print() end

lua> seq=sb_find(7)
lua> loop(10,seq)
7 4801 9547 9601 11311 11317 11941 11953 13033 13327

lua> seq=sb_find(31)
lua> loop(10,seq)
31 619 709 739 769 829 859 919 1549 1579
```

EMAIL PM FIND QUOTE REPORT

04-14-2019, 12:27 AM **Post: #36**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, **Bernd Grubert** and **Albert Chan**:

> **Bernd Grubert Wrote:** ⇨ (04-13-2019 09:01 PM)
>
> I don't understand the term ***composite*** in the context of ***Tier 2***. [...] Please explain what's meant by *composite*.

With pleasure. In this context ***composite*** simply means ***not prime***, i.e., if a number is not prime (thus it can be factored as the product of at least two not necessarily distinct prime factors) then it is considered ***composite***. For instance:

25 is ***composite*** because it's ***not** a **prime***, as it can be factored as **5 * 5** (two identical *prime* factors).

23 isn't ***composite*** because it's *a* ***prime***, as its prime factoring is just itself, **23** (a single prime).

Thanks for your interest. Should you have any further doubts, just tell me.

> **Albert Chan Wrote:** (04-13-2019 11:35 PM)
>
> I recently created nextprime.lua, which is needed for solving Tier 2 puzzle.
> [...]
> lua> seq=sb_find(***7***)
> lua> loop(10,seq)
> ***7 4801 9547 9601 11311 11317 11941 11953 13033 13327***

Nope, this computed sequence for base 7 and all others that follow are *incorrect* and thus not valid solutions for ***Tier 2***. I think you misunderstood what's actually being asked, which I repeat here with some relevant highlighting for your convenience:

- *"Write a program that accepts a base B (2 to 36) and outputs in order those **prime** numbers N such that* $S_B(N)$ *is **composite** and **distinct** from the previous ones, where* $S_B(N)$ *is a function which returns the sum of the base-B digits of a given integer N."*

Best regards to all.
V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

04-14-2019, 05:39 PM **Post: #37**

Bernd Grubert
Member

Posts: 91
Joined: Dec 2013

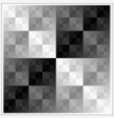
RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hello Valentin,
Thanks for the explanation. Now everything is clear.

Best regards
Bernd

EMAIL PM FIND QUOTE REPORT

04-14-2019, 08:57 PM **Post: #38**



John Keith
Senior Member

Posts: 615
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Somehow I had completely missed Tier 2 until I saw Bernd's post #35. Then I thought I had a good program until I saw Albert's reply and realized the uniqueness requirement, so back to the drawing board.

This problem turns out to be a good fit for the 50g and the Prime, both of which have **NEXTPRIME** and **ISPRIME?** as built-in functions.

My program also uses the **I->BL** command plus a couple of other commands from ListExt. I have tried to keep *stackrobatics* to a minimum in the interest of readability.

```
%%HP: T(3)A(R)F(.);
\<< I\->R \-> b n
\<< { } 1 1. n
START NEXTPRIME DUP b I\->BL LSUM DUP
IF ISPRIME?
THEN DROP
ELSE ROT SWAP DUP2
IF POS
THEN DROP SWAP
ELSE + OVER + SWAP
END
END
```

NEXT DROP DUP SIZE 2. / LDIST EVAL
\>>
\>>

Inputs are the base on level 2 and the number of primes to check on level 1. Output are two separate lists, the composites and the primes.

I would classify the size (163 bytes) and speed as reasonable if not exactly prize-winning, and it is sort of cheating as it uses so many pre-existing commands. I shudder to think of writing such a program on a "classic" era machine.

I have checked the first 100000 primes for 7 and 31, which take over 5 minutes each on the emulator, so my results are nowhere near as extensive as Albert's. Still a neat problem, I only wish I had noticed it earlier.

EMAIL PM FIND QUOTE REPORT

04-20-2019, 11:47 AM (This post was last modified: 04-20-2019 11:48 AM by Bernd Grubert.) **Post: #39**

Bernd Grubert
Member

Posts: 91
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hello Valentin,
here is my solution to Tier 2. It is 192 bytes long, due to the lack of prime number checking and the remainder function on the HP-15C.
I have done the test runs on the HP-15C emulator on a PC, since the processing time on my DM-15L is far too long...

Since the largest integer number the HP-15C can exactly represent is 9,999,999,999. , this implementation of the Miller-Rabin algorithm can check only number up to 99,999.
Due to memory limitations, on the real HP-15C and the DM 15L the longest sequence is 26 values.

For base 31 I got the sequence: 619, 18257, ...,(I stopped at 34139 after ~90 min., because I didn't want to wait any longer)

For base 7 I got the sequence: 4801, ...,(I stopped at 23451 after ~60 min.)

I have attached an HTML-documentation and a txt-file, that can be read into the emulator after changing the extension back to ".15c":
Tier_2.htm (Size: 49.27 KB / Downloads: 1) and
Tier_2.txt (Size: 6.5 KB / Downloads: 2) .

Best regards
Bernd

EMAIL PM FIND QUOTE REPORT

04-21-2019, 08:06 AM (This post was last modified: 04-21-2019 11:47 PM by Gilles.) **Post: #40**

Gilles
Member

Posts: 171
Joined: Oct 2014

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Tier 1 :

Here is my solution without reading others responses. I image that there exists better way. This one is "bestial" ;D
Always impressed how fast NewRPL is.

Brutal force :

1/ HP50g NewRPL or RPL

Code:

```
«
 0
 1000001111 1E10 FOR 'n'
  n ->STR
  IF "0" "" SREPL 1 == THEN
   IF "1" "" SREPL 1 == THEN
    IF "2" "" SREPL 1 == THEN
     IF "3" "" SREPL 1 == THEN
      IF "4" "" SREPL 1 == THEN
       IF "5" "" SREPL 1 == THEN
        IF "6" "" SREPL 1 == THEN
```

Solved in **only 1.3s in newRPL** (on my PC) , **116s with HP50g hdw**, much much slower **in 779s in RPL** (on my PC with Emu48). NewRPL 600 times faster in this case on a PC.
2/ HP50g RPL with ListExt, shorter but slower

Code:

```
« 0 1000001111 1E10 FOR n  n I->NL LDDUP SIZE 10 == { 1 + } IFT  11111 STEP »
```

EMAIL PM FIND QUOTE REPORT

04-24-2019, 06:36 AM **Post: #41**

Massimo Gnerucci
Senior Member

Posts: 2,152
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Albert Chan Wrote:** (04-23-2019 05:59 PM)
>
> sb_find(), version 6. Switched to Python to extend search range.
> As expected, Python code is even shorter.

Hi Albert, really interesting (and a *little* beyond my skills), as usual, but wasn't this one of Valentin's rules? :)

> **Valentin Albillo Wrote:** (03-21-2019 03:08 AM)
>
> **Rules:**
> [list]
> [*]Using anything other than a **physical** or **emulated HP calculator** is strictly disallowed. Also **no** *VBA*, *Excel*. *Pascal*, *C/C#/C++*, *Java*, *Python*, *Haskell*, etc. code, please go elsewhere for that. You must write your code in a language supported in some HP calc (i.e.: ***RPN***, ***RPL***, ***71B BASIC***/***FORTH***, etc).

Have a nice day!

Greetings,
Massimo

-+×÷ ↔ *left is right and right is wrong*

PM WWW FIND QUOTE REPORT

04-24-2019, 08:57 PM **Post: #42**

Valentin Albillo
Senior Member

Posts: 636
Joined: Feb 2015
Warning Level: 0%

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, Bernd Grubert, Gilles and Massimo Gnerucci:

> **Bernd Grubert Wrote:** (04-20-2019 11:47 AM)
>
> Here is my solution to **Tier 2**. It is 192 bytes long, due to the lack of prime number checking and the remainder function on the **HP-15C**. [...] I have done the test runs on the **HP-15C** emulator on a PC, since the processing time on my DM-**15**L is far too long... Since the largest integer number the **HP-15C** can exactly represent is 9,999,999,999. , this implementation of the Miller-Rabin algorithm can check only number up to 99,999. Due to memory limitations, on the real **HP-15C** and the DM **15**L the longest sequence is 26 values. [...] For base 31 I got the sequence: 619, 18257, ...,(I stopped at 34139 after ~90 min., because I didn't want to wait any longer) [...] For base 7 I got the sequence: 4801, ...,(I stopped at 23451 after ~60 min.) [...] I have attached an HTML-documentation and a txt-file, that can be read into the emulator after changing the extension back to "**.15c**"

Thanks a lot for your interest and for your nice solution, **Bernd**, much appreciated !

Matter of fact, I do appreciate your solution over any purported "solutions" written in non-HP calc languages/environments, such as ***Mathematica***, ***Lua*** and ***Python***, which I expressly said in my OP that **should not be used at all** but that *certain individual who routinely disregards the rules* did nevertheless use, as I discuss below.

You, on the other hand, did abide by the rules and took the trouble to use an actual **HP calculator** (or emulator) and wote actual ***RPN* code**, fearlessly wrestling with its limitations, to produce an actual solution to the challenge I posted, instead of going the lazy route of using high-level languages on a full-fledged PC, which lacks any merit whatsoever and which for me amounts to **trolling**.

So, again, **Bernd**, thank you very much for your valuable contribution, my challenges are created for people like you who work hard on them to produce solutions *under the constraints given*, thus fulfilling my stated *purpose*, which is to have people using their **HP calculators**, with their limitations and warts and all, ***not*** using some fancy non-HP languages and/or environments, which completely defeats the purpose.

> **Gilles Wrote:** (04-21-2019 08:06 AM)
>
> **Tier 1:** Here is my solution without reading others responses. I image that there exists better way. This one is "bestial" ;D Always impressed how fast New**RPL** is. Brutal force :
>
> 1/ **HP50g** New**RPL** or **RPL** [...] Solved in only 1.3s in new**RPL** (on my PC) , 116s with **HP50g** hdw, much much slower in 779s in **RPL** (on my PC with **Emu48**). New**RPL** 600 times faster in this case on a PC. [...] 2/ **HP50g RPL** with ListExt, shorter but slower [...]

Thanks a lot for your **RPL**/New**RPL** solutions, **Gilles**, much appreciated. What I told **Bernd** above also applies equally to you so for the sake of brevity I won't repeat it here.

Again, thanks for your interest and for your time, hope you enjoyed the challenge as I certainly enjoyed your solutions, keep them coming for future ones !

> **Massimo Gnerucci Wrote:** (04-24-2019 06:36 AM)
>
> > **Albert Chan Wrote:** (04-23-2019 05:59 PM)
> >
> > Switched to **Python** to extend search range. As expected, **Python** code is even shorter.
>
> Hi Albert, really interesting (and a *little* beyond my skills), as usual, but **wasn't this one of Valentin's rules?**
>
> > **Valentin Albillo Wrote:** (03-21-2019 03:08 AM)
> >
> > Using anything other than a **physical** or **emulated HP calculator** is **strictly disallowed**. Also **no** *VBA*, *Excel*. *Pascal*, *C/C#/C++*, *Java*, *Python*, *Haskell*, etc. code, please go elsewhere for that. You **must** write your code in a language supported in some HP calc (i.e.: ***RPN***, ***RPL***, ***71B BASIC***/***FORTH***, etc).

Thanks for pointing this out, **Massimo**, I didn't read Mr. Chan's posts because I've placed him in my *Ignore* list so that I don't read his post anymore, as he has shown an utter **disregard** for the rules I so clearly state in my challenges, thus completely **defeating** the purpose and probably ruining them for others, which I find profoundly **disrespectful**.

To wit:

- *The purpose of my challenges is to offer HP-fan fellow readers the opportunity to actually get to use their HP calculators and their languages to solve allegedly interesting math topics, so that perhaps we all learn some new exciting math facts and some worthwhile HP-calc programming techniques which are cleverly used to overcome the natural limitations of our beloved HP calcs and their languages. Nothing more and nothing else.*

If some disrespectful individual like Mr. Chan then goes on and *completely ignores* the requirement to use HP calcs and their languages (***RPN***, ***RPL***, ***71BASIC***, ***71FORTH***, ***PPL***, ***Saturn*** *assembler*, etc) and uses instead *exclusively* such software as ***Mathematica***, ***Lua***, ***Python*** or whatever on a PC to effortlessly overcome the aforementioned natural limitations of our calcs and provide almost-instant solutions, then:

- **No HP calcs/languages are used at all**, which completely *defeats* the intended purpose.
- **No HP calcs/languages limitations are overcome at all**, with was the idea, as the challenges' difficulty is geared to HP calc/languages, not to *Mathematica*/*Lua*/*Python* running on a PC where the difficulty and inconvenience are diminished by orders of magnitude, which amounts to *shameful cheating*.
- **No HP calcs/languages' new interesting programming techniques are created** for everyone to learn, which again completely *defeats* the intended purpose.

So, what this individual, Mr. Chan, is continuously doing amounts to:

- Utter **disrespect** to the rules I explicitly stated, which aren't arbitrary but do have the explicit intended purposes stated above.
- Shameful **cheating**, as he is using software/harware orders of magnitude more powerful that the intended one, namely HP calcs/languages.

- **Spoiling**, as he gives lots of versions and explicit solutions with no "spoilers" warnings whatsoever, thus spoiling the fun for any rule-abiding forum readers, and probably discouraging them from even attempting the challenge using their HP calculators.

- **Trolling**, as he has done this any number of times by now. He did that with my *cin(x)* challenge, for example, and with all the recent ones. He does it all the time though he could certainly go on and post his ***Mathematica***/***Lua***/***Python*** code in *Mathematica*/*Lua*/*Python* forums (as he did with my *Cin(x)* challenge; but he nevertheless posted his **Lua** or **Python** code here as well).

In one of my challenges he said something to the effect that he only has an *HP-12C* at hand but that is hardly a valid excuse as he's obviously using a PC or other device to run his *Mathematica*/*Lua*/*Python* code and there are any number of ***free*** emulators of HP calcs available for Windows, *Mac*, *Android* and *iOS*, such as ***Free42***, ***V41***, ***Emu71/DOS***, ***Emu71/Win***, ***HP-25***, most ***RPL*** models, ***Prime***, etc. etc., which he could easily use but he **doesn't**.

This being so, I've placed Mr. Chan in my *Ignore* list and won't read or comment on his solutions. I'm totally fed up with his continued **cheating**, **trolling** and **disrespect**.

If he's so keen on showing off his abilities he should go and post his *Lua* marvels in the *Lua* forums and his *Python* achievements in the *Python* forums, not in my challenge threads, or at the very least be an sportman and provide *first* an acceptable solution using HP calcs/languages, as per the rules. At any rate, if he goes on trolling my challenges here like that, I'll certainly take appropriate measures.

Best regards to all.
V.

Find All My HP-related Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

04-24-2019, 09:30 PM (This post was last modified: 05-13-2019 07:32 PM by Albert Chan.) **Post: #43**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, Valentin Albillo

I did not know people were offended.
I had deleted all posts since your last posting.

Sorry about that ...

I did explain my HP-12C can't handle primes.
I am learning the emulation software, but that take some time ...
Tried FPRIM(n) in emu71, but only gives "Err: Invalid Expr"

This is from your last PM to me:

> **Valentin Albillo Wrote:**
>
> Hi, **Albert Chan**:
>
> Your second attempt at ***Tier 2*** is essentially correct but for a minor detail. You posted (for base ***7***):
>
> ```
> lua> sb_find(7)
> 7      1 <<<<<<<<<<<<<<<
> 4801      25
> 201683      35
> 16470859      49
> 115296019      55
> ```
>
> but your first result above (**7 1**) is *incorrect* because **1** is neither a *prime* nor a *composite*, it's a *unit*. Not being a composite, as asked, makes it an invalid result which perhaps you might want to correct in your post before I post my solutions.
>
> Thanks for your interest and your clear & concise solutions, best regards.
> V.
> .

Update:
Replaced all my Python code with XCas, so HP Prime user can try out.
The 1 Mathematica post and Lua code got quoted by others, so I felt better leave it alone.
At the time, I was too excited cin puzzle is solved in my head ...

EMAIL PM FIND QUOTE REPORT

04-26-2019, 07:13 PM (This post was last modified: 04-27-2019 01:25 AM by Albert Chan.) **Post: #44**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

I did downloaded the **Emu71/Windows**, but unable to get the code fully worked out.

HP-71B BASIC code below only print out possible sb candidates.
The program cannot be completed without ISPRIME(), so I never posted it.

What it does is output values of permutations of base-B digits that have the inputed sb value.

Quote:

```
10 INPUT "BASE ?"; B
20 INPUT "DIGITS ?"; N
30 DIM S(N),L(N),H(N)
40 INPUT "SB VALUE ?"; S(N)
50 I = N
60 GOSUB 100
70 END

100 L(I) = S(I)-(B-1)*(I-1)
110 IF L(I) < 0 THEN L(I) = 0
120 H(I) = S(I)
130 IF H(I) >= B THEN H(I) = B-1

150 IF L(I) > H(I) THEN RETURN
160 IF I = 1 THEN GOTO 200
170 S(I-1) = S(I)-L(I)
180 I = I-1 @ GOSUB 100 @ I = I+1
190 L(I) = L(I)+1 @ GOTO 150

200 T = 0
210 FOR K = N TO 1 STEP -1
220 T = T*B+L(K)
230 NEXT K
240 PRINT T;

250 INPUT "MORE?";Y
260 IF Y<>0 THEN RETURN
```

Example: for base-7, upto 12 decimal digits => 15 base-7 digits

```
>RUN
>BASE ? 7
>DIGITS ? 15
>SB VALUE ? 65
```

1694851493 → 1936973135 → **1971561941** (first prime)

```
>RUN
>BASE ? 7
>DIGITS ? 15
>SB VALUE ? 77
```

83047723205 → 94911683663 → 96606535157 → **96848656799** (first prime)

BTW, where to get ISPRIME() (or equivalent) for the HP-71B emulator ?

Edit: change PRINT "X?" @ INPUT X to INPUT "X?";X

EMAIL PM FIND QUOTE REPORT

04-26-2019, 10:50 PM (This post was last modified: 04-26-2019 10:53 PM by rprosperi.) **Post: #45**

rprosperi
Senior Member

Posts: 4,439
Joined: Dec 2013

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Albert Chan Wrote:** (04-26-2019 07:13 PM)
> I did downloaded the **Emu71/Windows**, but unable to get the code fully worked out.
>
> BTW, where to get ISPRIME() (or equivalent) for the HP-71B emulator ?

Congratulations Albert, on upgrading to EMU71! I think you will find much more participation by others with your posts if they include HP code. You happen to have made a good choice as the 71B is my favorite machine, so I can answer many questions about using it, but I can't contribute much regarding a lot of the math you frequently post.

For ISPRIME(), I don't have a LEX file with this, however the PRIMLEX LEX file (PRIM(X) returns the lowest Prime factor of X) found on this page may meet the need.

Also, to input N with a prompt, use this:

100 INPUT "What is N?"; N

--Bob Prosperi

EMAIL PM FIND QUOTE REPORT

04-27-2019, 12:36 AM (This post was last modified: 04-27-2019 12:45 AM by Gilles.) **Post: #46**

Gilles
Member

Posts: 171
Joined: Oct 2014

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

> **Valentin Albillo Wrote:** (03-28-2019 01:38 AM)
> ## Tier 1 - The Challenge:
> (...)
>
> Also, being a 10-digit number and having all its digits distinct means that its digits are *0, 1, 2, 3, ... , 9* in some order and thus their sum is *1 + 2 + 3 + ... + 9 = 45*, which is divisible by **9** so each *Homage* number has to be divisible by 9 too. As 9 is coprime to 41 and 271, each *Homage* number N must be divisible by their product, i.e., by *41 * 271 * 9 =* ***99,999***.

Good catch the divisibility by 9. I totaly missed this.

EMAIL PM FIND QUOTE REPORT

04-28-2019, 12:08 PM (This post was last modified: 07-23-2019 02:10 PM by Albert Chan.) **Post: #47**

Albert Chan
Senior Member

Posts: 1,226
Joined: Jul 2018

RE: [VA] Short & Sweet Math Challenge #24: "2019 Spring Special 5-tier"

Hi, rprosperi

Thanks for the tip. I finally get **FPRIM** working, but had to switched to **EMU71/DOS**
This is my updated HP71B BASIC listing.

> **Quote:**

```
10 ON ERROR GOTO 190
20 INPUT "BASE?";B
30 N=CEIL(12/LOG10(B)) @ DIM S(N),L(N),H(N)
40 FOR C=4 TO (B-1)*N
50 IF GCD(C,B-1)=1 AND FPRIM(C,C)=0 THEN
60 T=0 @ I=N @ S(I)=C @ GOSUB 100 @ PRINT C;T
70 END IF
80 NEXT C
90 END

100 L(I)=MAX(S(I)-(B-1)*(I-1), 0)
110 H(I)=MIN(S(I), B-1)
120 IF L(I)>H(I) OR T THEN RETURN
```

```
130 IF I=1 THEN GOTO 170
140 S(I-1)=S(I)-L(I)
150 I=I-1 @ GOSUB 100 @ I=I+1
160 L(I)=L(I)+1 @ GOTO 120

170 FOR K=N TO 1 STEP -1 @ T=T*B+L(K) @ NEXT K
180 T=FPRIM(T,T) @ RETURN
190 T=-T @ RETURN
```

>RUN
BASE? 7
25 4801
35 201683
49 16470859
55 115296019
65 1971561941
77 96848656799
85 -1.3564461457E12

>RUN
BASE? 31
49 619
77 18257
91 59581
119 1787459
121 2769601
133 13851853
143 22164503
161 372178931
169 629810569
187 7987533097
203 23073248663
209 54109095389
217 247613526037
221 357635354291
247 -6.82312829953E12
253 -1.19404745241E13
259 -1.70578207488E13

Ignore lines with negative numbers. It just meant T (for the sb value) overflow 12 digits.

Note: It is possible primes not in sorted order. Example:
>RUN
BASE? 2
4 23
6 311
8 383
...
36 206141652991
38 -1.01361228185E12 <- actual T = 1030791102463
39 824633720831
40 -1.09951162777E12 <- actual T = 2196875771903