HP Forum Archive 15

[Return to Index | Top of Index]

Short & Sweet Math Challenge #13: Adding up to infinity
*Message #1 Posted by **Valentin Albillo** on 30 Jan 2006, 6:39 a.m.*

Hi all,

Just back from Xmas holidays, let's begin the year with a brand new S&SMC, this time lucky number #13, which though not particularly difficult, it's slightly 'busier' than usual so, to compensate for this, it comes with a 'prize' attached for all of you who can deliver, details at the end of this post.

Caveat emptor: since this is S&SMC #**13** you're hereby warned that failure to *at least try* will be considered disrespect to the number 13 and thus you'll suffer *a whole year of pretty bad luck*. You've been warned. :-)

The Challenge

You must write a program for your favorite HP (preferably) calc, which shall do the following:

1. For each one of the 8 infinite sums given below, it must compute *a numerical value S for the sum*. A correct value will be rewarded with 1 point.

2. Then, it must convert this value S to *rational form* and output the numerator N and denominator D of the resulting fraction, N/D, such that this fraction is in lowest terms, and is as simple as possible while correctly evaluating to S, allowing for slight rounding inaccuracies in the computed value S. For instance, if you obtain S = .666666666668, you're expected to produce **2/3**, *not* 166666666667/250000000000. A correct value will be rewarded with another point.

3. You must then state whether your resulting fraction is indeed the *exact* value of the infinite sum, in your opinion, or just an approximation, and in this last case, to how many digits do you think it is correct. If you're right, you'll be rewarded with yet another point.

   Example: Suppose you're asked to compute SUM(N = 1, N -> Inf, f(N)/2^N) where f(N) = N/3.

   First, your program finds that S = .666666666668, approximately. Then, your program converts this value to a fraction as stated, giving S = 2/3 And finally you confirm that, in your opinion, the exact value of S is indeed 2/3. All are correct, so you would get 3 points for this sample case.

Notice that despite there being eight different sums to compute, *they are similar enough that your program will be nearly the same in all cases*, you just need to additionally code the particular function being summed up. Of course, the code to convert to fractional form is the exact very same for all eight cases, and last but not least, your expert judgment about what the exact result should be requires no coding at all, just smarts and/or intuition on your part.

To give an idea of the magnitude of the task ahead, all in all, and supposing you were using an HP-71B for example, you should be able to code the infinite sum calculation in a couple of lines, the particular function being summed up in another two lines at most, and the real-to-fraction routine in some five lines, for a total of 9-10 lines of code for the very first sum, and 2-3 additional lines for each remaining case. Not much, if you know how to do it.

The particular infinite sums to compute for this challenge are the following:

1. **S = SUM(N=1, N -> Inf, f(2^N)/2^N)**

   where f(N) counts the number of *odd digits* (i.e.: 1,3,5,7,9) in N

   for example: f(2048) = 0, f(2718281828) = 3, f(3141592654) = 6

2. **Same as above but f(N) counts instead the number of *even* digits (i.e.: 0,2,4,6,8) in N**

   for example: f(2048) = 4, f(2718281828) = 7, f(3141592654) = 4

3. **Same as above, but f(N) = 1 if N has an *even* number of digits and 0 otherwise.**

```
          for example:  f(2048) = 1, because 2048 has 4 digits and 4 is even
                        f(777)  = 0, because  777 has 3 digits and 3 is odd
```

4. **Same as above, but f(N) counts the number of *odd* digits in *odd* places in the decimal expansion of N**

(the 1st digit is the 1st digit to the left of the decimal point).

```
    For example: f(901) = 2,
                 f(210) = 0, (the only odd digit, 1, is at an even place (2nd) )
                 f(811) = 1
```

- **S = SUM(N=1, N-> Inf, f(N)/10^N)**

  where f(N) is the same as in the previous sum above.

- **S = SUM(N=1, N-> Inf, f(N)/2^N)**

  where f(N) = Integer part of (Pi/2 * N)

- **S = (SUM(N=1, N-> Inf, f(N))/5+1/10)^2**

  where f(N) = e^(-N*N/100)

(In this case you must *not* output a fraction but simply the decimal value of S, and must try and guess what the exact value of S is)

- **S = SUM(N=1, N -> Inf, f(N)/ 5^N)**

  where f(N) = Integer part of (N*e^(Pi/3*Sqrt(163)))

That makes a maximum of **23 points** if you correctly compute, convert to fractional form, and guess the exact value of the sum for all 8 cases.

The prize

Everyone whose solutions get **15 points** or more will be entitled to receive by e-mail the following, still *unavailable on-line* Datafile articles of mine, in PDF format, namely:

- **HP-71B Math ROM Baker's Dozen Vol. 2**

  Multidimensional integrals, solving systems of non-linear equations, computing all eigenvalues real and complex of arbitrary square matrices, and more

- **HP-71B Fantastic FOUR**

  Multiplying very large numbers fast by means of Fast Fourier Transforms

- **HP-71B Short & Sweet Sudoku Solver**

  Automatically solves any sudoku puzzle without user's intervention, by using advanced techniques such as recursion, bitboards, and boolean operations.

- **HP-71B Sudoku Solver Sublime Sequel**

  Improved version of my Sudoku Solver, incorporating new criteria for much faster results. Includes Test Suite of 15 choice Sudoku puzzles, from the simplest to the most difficult.

- **Mean Matrices**

Introducing a series of simple and small integer-valued randomly-generated matrices that nevertheless are nearly intractable by conventional means due to their extreme ill-conditioning. Several cases are discussed at length, with (discouraging) results for the 71B and exact results with Mathematica.

- **HP-71B Minimax Polynomial Fit**

  Computes the minimax polynomial fit to a given set of datapoints, a given function in an interval, or a dataset read from a file. The user can specify either the degree of the desired polynomial or else the maximum absolute error over the given dataset or interval and let the program compute the lowest-degree polynomial that meets the requirement.

  The resulting polynomial can be evaluated at any point, scanned for its maximum error over a given interval, and is guaranteed to have an absolute maximum error lower than any other polynomial fit, least-squares included.

The usual caveats apply, namely: (1) Do not post just solutions, actual code is mandatory and (2) Code must be for an HP (preferably) calculator; posting code written in Visual Basic, Java, FORTRAN, or any other PC language will be considered unpolite and disrespectful.

That's all. Hoping you'll like this challenge and waiting for your inputs,

Best regards from V.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #2 Posted by **Mike T.** on 30 Jan 2006, 12:42 p.m.,*
in response to message #1 by Valentin Albillo

OK - I don't think I've got enough space left in my HP33C to actually solve any of these at the same time but when (if) I get round to it I could use the following to convert decimal fractions to rational form.

Unfortunatly I wouldn't get a point every time as I've just found a bug, but I think it is worth posting anyway as I probably won't get round to fixing it! I think it only affects the conversion when the 10th digit in the decimal value is greater than 5 - but it is obvious when it happens, so it is good enough for now. If you can fix it (on an HP33C) post you version here!

```
001 23 3     STO 3
002 15 32    g INT
003 23 2     STO 2
004 24 3     RCL 3
005 15 33    g FRAC
006 23 0     STO 0
007 0        0
008 23 1     STO 1
009 24 0     RCL 0
010 15 71    X=0
011 13 24    GTO 24
012 1        1
013 23 51 1  STO + 1
014 24 1     RCL 1
015 24 0     RCL 0
016 71       /
017 15 33    g FRAC
018 15 61    X<>0
019 13 12    GTO 12
020 24 1     RCL 1
021 24 0     RCL 0
022 71       /
023 23 0     STO 0
024 24 2     RCL 2
025 24 1     RCL 1
026 24 0     RCL 0
027 13 00    GTO 00
028 15 71    X=0
029 13 32    GTO 32
030 71       /
031 15 12    g RTN
032 22       R
033 15 12    g RTN


Uses R0, R1, R2, R3
```

```
<f PRGM>
Enter decimal number <R/S>
Denominator <R> (Round down)
Numerator <R>
Integer


eg
<f PRGM>
0.5 <R/S>
2 <R>
1 <R>
0


0.5 = 1/2 !!
```

Enjoy.

Mike T.

Edited: 1 Feb 2006, 7:07 a.m.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #3 Posted by **Antoine M. Couëtte** on 30 Jan 2006, 4:06 p.m.,*
in response to message #1 by Valentin Albillo

Jan 30, 2005

Valentin,

Out of topic information request ... although indirectly related to your Challenge #13 post.

Given its features, I would guess that your "HP-71B Minimax Polynomial Fit" uses Chebyshev Polynomials Approximation (CPA) . CPA needs samples "evenly spaced on a circle", i.e. a more dense sampling at both ends of interval - sometimes a cumbersome constraint - BUT CPA achieves the smallest absolute error for a given degree of polynomial.

On the other hand FFT approximation simply needs evenly spaced samples BUT is generally NOT accurate and may give big (sometimes wildly varying) errors at interval ends.

I need to approximate a function which is known at evenly spaced points. I do not want to use FFT approximation because of its (gross) unreliability and unpredictability at interval ends. Could I easily use your "HP-71B Minimax Polynomial Fit" with evenly spaced samples without the need to (linearly) interpolate function values at the required "evenly spaced on a circle" points?

Thank you for your Kind Attention

Antoine

antoine.m.couette@club-internet.fr

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #4 Posted by **Valentin Albillo** on 31 Jan 2006, 7:01 a.m.,*
in response to message #3 by Antoine M. Couëtte

Hi, Antoine:

Antoine wrote:

"Given its features, I would guess that your "HP-71B Minimax Polynomial Fit" uses Chebyshev Polynomials Approximation (CPA)"

You'd guess wrong, I don't use CPA. My program computes the *exact* minimax polynomial for the case of a given arbitrary dataset (x_i, y_i), where you can either select the degree of the minimax polynomial beforehand, or else specify a maximum absolute error and let the program select the lowest-degree polynomial that complies.

For the case of some user-specified function f(x) in a given interval (which can be *anything*: a built-in function, an user-defined function, a function in some external ROM, including Math ROM's 'funny functions' Solve and Integrate, etc.), the function itself is firstly used to generate a discrete dataset, then my program fits an exact minimax polynomial to that dataset, which will very closely approximate the true minimax polynomial for the continuous function. Also, you can improve the accuracy of the approximation by simply generating more data points in the interval.

"I need to approximate a function which is known at evenly spaced points [...] Could I easily use your "HP-71B Minimax Polynomial Fit" with evenly spaced samples without the need to (linearly) interpolate function values at the required "evenly spaced on a circle" points?"

Certainly. As stated, my program doesn't use CPA and it works with any arbitrary dataset as long as the x ordinates are in increasing value and there are no two equal x ordinates. Whether they're evenly spaced or not is irrelevant and, matter of fact, when a user-specified function f(x) is supplied, the program generates *evenly spaced* datapoints from it and fits the polynomial to that.

Also, as discussed in some past posts (search for them in this forum), it's very easy to use my program to compute minimax polynomials having some desired peculiarity, such as only using odd or even powers of x, or forcing the coefficient of x to be exactly 1, say, etc.

Hope this answers your questions. Thanks for your interest, and

Best regards from V.

Re: Short & Sweet Math Challenge #13: Adding up to infinity
*Message #5 Posted by **Antoine M. Couëtte** on 2 Feb 2006, 5:01 a.m.,*
in response to message #4 by Valentin Albillo

Hello Valentin,

Thank you for your kind, courteous and extremely interesting reply.

Given your additionnal detailed information, I would keep thinking that your mathematical approach in your "HP-71B Minimax Polynomial Fit" is probably close in essence to the logic of Chebyshev Polynomials Approximation (CPA). Both methods claim that for a given polynomial degree, their maximum absolute error is the smallest one you can get ...

By the way, do you have a similar program for the HP41? I did write one for the HP41 which works very well for my use, with the constraints of having to interpolate function at specific sampling points "equally spaced on a circle".

I am willing to communicate more on this with you in order to compare my results on HP41X (you have HP41X from Hrast don't you ?) with yours obtained on your "HP-71B Minimax Polynomial Fit". However this forum is not an appropriate place to do so.

Would you be so kind as to send me an e-mail at my e-mail address " antoine.m.couette@club-internet.fr " so that I can give your more details on the function I am studying and on the way I currently represent it with CPA.

Best Regards from

Antoine

Re: Short & Sweet Math Challenge #13: Adding up to infinity
*Message #6 Posted by **Valentin Albillo** on 2 Feb 2006, 6:21 a.m.,*
in response to message #5 by Antoine M. Couëtte

Hi again, Antoine:

Antoine wrote:

"Thank you for your kind, courteous and extremely interesting reply."

You're welcome.

"Given your additionnal detailed information, I would keep thinking that your mathematical approach in your "HP-71B Minimax Polynomial Fit" is probably close in essence to the logic of Chebyshev Polynomials Approximation"

I don't think so, but anyway, each and every method that deals with the computation of minimax polynomials must address the same issues nevertheless, so they all share similar logics, in essence.

But where CPA deals with an special kind of orthogonal polynomials with some weighting function involved, and then uses their equiripple properties to approximate true minimax, I do nothing of the kind, and I do not find approximate minimax polynomials but exact ones, to the limits imposed by finite accuracy arithmetic.

"By the way, do you have a similar program for the HP41?"

No, because my approach is very computationally intensive and requires a large amount of RAM, so the HP-71B is about the minimal platform where it would run acceptably. The HP-41C would be far too slow and limited except for low degrees and small datasets.

"I am willing to communicate more on this with you in order to compare my results on HP41X (you have HP41X from Hrast don't you ?) with yours obtained on your "HP-71B Minimax Polynomial Fit".

I'm willing to obligue but I don't have HP41X because I don't have any 48/49 models, matter of fact I can't stand RPL machines. Not to discourage others from liking or using them, of course, to each his own.

"However this forum is not an appropriate place to do so. Would you be so kind as to send me an e-mail at my e-mail address " antoine.m.couette@club-internet.fr " so that I can give your more details on the function I am studying and on the way I currently represent it with CPA."

I do not concur in this forum not being an appropriate place, as the topic is mathematical in nature ("Terms of Use") and does involve vintage HP machines (HP-41C and HP-71B), so it certainly belongs here and might well be of educational or practical use to others, but if there are any confidentiality matters involved or you aren't comfortable with discussing it here, you can e-mail me at my public e-mail address, which you can find at my web site http://membres.lycos.fr/albillo/calcmain.htm

Best regards from V.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #7 Posted by **PeterP** on 31 Jan 2006, 11:25 p.m.,*
in response to message #1 by Valentin Albillo

Valentin,

First, thanks for placing to us another challenge that will help us fill our day with more things that are fun and less things that put bread on the table ;-)

Secondly, the first sum is written as Sum(f(2^n)/2^n), for n=1 to infinity) yet then you go on to say f(N) counts the number of odd digits. So I presume - yet might be totally wrong - that you actually mean the first sum to be Sum(f(N)/2^N), for n = 1 to infinity. Not that I would have any better idea how to do this than what is written, just want to make sure I start on the right track...

Cheers

Peter

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #8 Posted by **Valentin Albillo** on 1 Feb 2006, 4:38 a.m.,*
in response to message #7 by PeterP

Hi, PeterP:

PeterP wrote:

"First, thanks for placing to us another challenge that will help us fill our day with more things that are fun and less things that put bread on the table ;-)"

You're welcome. It doesn't help me put bread on my table either, to spend time concocting these challenges ;-)

"Secondly, the first sum is written as Sum(f(2^n)/2^n), for n=1 to infinity) yet then you go on to say f(N) counts the number of odd digits. So I presume - yet might be totally wrong - that you actually mean the first sum to be Sum(f(N)/2^N), for n = 1 to infinity. Not that I would have any better idea how to do this than what is written, just want to make sure I start on the right track..."

Thanks for your interest and you've done the right thing by first clearing out any doubts you had but no, you're indeed *wrong* and the sum is exactly as written, i.e.:

```
  n->oo
 -------
```

```
       \
        >      f(2^N)/2^N = f(2^1)/2^1 + f(2^2)/2^2 + f(2^3)/2^3 + ...
       /
      -------             = f(2)/2 + f(4)/4 + f(8)/8 + f(16)/16 + ...
       n = 1
                          = 0/2 + 0/4 + 0/8 + 1/16 + 1/32 + 0/64 + ...


       as f(N) counts the number of odd digits (i.e.: 1,3,5,7,9) in N,
       so we have:


              f(2)  = 0    ("2" has no odd digits)
              f(4)  = 0    (ditto)
              f(8)  = 0    (ditto)
              f(16) = 1    ("16" has one odd digit, the "1")
              f(32) = 1    ("32" has one odd digit, the "3")
              f(64) = 0    ("64" has no odd digits)
              f(128)= 1    ("128" has one odd digit, the "1")
        etc.
```

I suggest you begin with sums 6, 7, and 8, which are easier, and then, overwhelmed by success, you go on to attack the remaining sums. I might even **increase** the prize by adding two other unavailable articles of mine, so there would be 8 articles for 8 sums.

As for the sums themselves, believe me this is *not* a boring, math-drudgery topic: the results are *awesome*, even surreal, you've never seen anything like this before !

Best regards from V.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #9 Posted by **Chris** on 1 Feb 2006, 7:12 a.m.,*
in response to message #8 by Valentin Albillo

```
Valentin


I have tried the problems with my HP-49G+ using for the main program 'CHALLENGE'
<<
  CLEAR
  'SIGMA(N=1,10000, F(2^N)/2^N)'
  EVAL ->Q
>>
where F(N) defines the function, SIGMA is the summation symbol. This outputs the results as fractions.


A typical F(N), fore example, for  the first problem is shown below and stored in 'F'


<<
  ->STR
  'A' STO
  0 'B' STO
  1 A SIZE 1 - FOR I
    IF A I I SUB NUM 2 MOD 1 == THEN
      1 'B' STO+
    END
  NEXT
  B
>>


The answers I have (some I am a bit dubious about or just do not recognise) are


1. 1
2. 4
3. 0
4. 1
5. 1/9
6. 91720561 / 343614632 = 2.66929133856 (?)
```

```
7. 3.14159265837 (PI)
8.200100/1 (?)


Even if the results are not quite correct I have enjoyed the Challenge.


Thank you very much


Chris
```

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #10 Posted by **Arnaud Amiel** on 1 Feb 2006, 7:30 a.m.,*
in response to message #9 by Chris

I only got (3) for now and I get 114/1025 but I am not fully sure. In that case F is << XPON 2 MOD >>

Need to work a bit more now

Arnaud

Re: Your score up to now

*Message #11 Posted by **Valentin Albillo** on 1 Feb 2006, 7:41 a.m.,*
in response to message #9 by Chris

Hi, Chris:

First of all, thanks for your interest in my challenge and your great solutions. Here's some preliminary score for you:

```
1. 1        Wrong value
2. 4        Wrong value
3. 0        Wrong value
4. 1        Wrong value
5. 1/9      Wrong value


6. 91720561 / 343614632 = 2.66929133856 (?)


            Correct value save a few ulps, Wrong fraction
            (remember what I stated about 2/3 vs 166666666667/250000000000)


7. 3.14159265837 (PI)


            Nearly-correct value but you can refine it


8.200100/1 (?)


            Correct value, Correct fraction
            (Your value is integer, so Denominator = 1)
```

You do not clearly state what you think are the correct values for the sum, i.e.: you may have gotten some result but think the real value is some other) so I can't accurately score that.

You might consider having a second look at it in order to correct/improve the wrong/improvable values and/or fractions, and come to that, specifically state what are the actual values in your opinion. I'm sure you can easily improve your score ! :-) Anyway, thanks a lot for your interest and

Best regards from V.

Re: Your score up to now

*Message #12 Posted by **J-F Garnier** on 1 Feb 2006, 3:45 p.m.,*
in response to message #11 by Valentin Albillo

Just a short and fast try to answer question #6 with the HP-71:

```
10 DEF FNF(N)=IP(PI/2*N)/2^N
20 S=0
30 FOR N=100 TO 1 STEP -1
40 S=S+FNF(N)
50 NEXT N
60 DISP S;FRAC$(S)
>
 2.66929133858 339/127
```

FRAC$ comming from the JPC ROM. (If HP48/49 owners can use the ->Q function, I may use the JPC ROM one?)

J-F

Edited: 1 Feb 2006, 4:05 p.m.

Re: Your score up to now

*Message #13 Posted by **Valentin Albillo** on 1 Feb 2006, 6:13 p.m.,*
in response to message #12 by J-F Garnier

Hi, Jean-François !

Jean-François wrote:

"Just a short and fast try to answer question #6 with the HP-71:[...] 2.66929133858 339/127"

> Correct value and correct fraction. Care to state whether you think this fraction is the exact value of the infinite sum ?

"FRAC$ comming from the JPC ROM. (If HP48/49 owners can use the ->Q function, I may use the JPC ROM one?)"

> Of course, and very clever on your part to remember that such a convenient function is included in the JPC ROM. Anyway, I'll provide a conversion routine using only standard keywords as part of my solution, for the benefit of interested users.

Come on, Jean-François, be a sport and post equally clever solutions for the remaining sums ... it's actually pretty easy ! :-)

Thanks for your interest and pretty solution, and

Best regards from V.

Re: Your score up to now

*Message #14 Posted by **Chris Dean** on 1 Feb 2006, 7:37 p.m.,*
in response to message #13 by Valentin Albillo

```
Valentin
I have made a few changes to my code and have had another go. The main program 'CHALLENGE'
<<
  CLEAR
  'SIGMA(N=1,10000, F(N)/2^N)'
  EVAL ->Q
>>
where F(N) defines the function, SIGMA is the summation symbol. This outputs the results as fractions.
A typical F(N), for example, for  the first problem is shown below and stored in 'F'
<< 2 ^
  ->STR
  'A' STO
```

```
  0 'B' STO
  1 A SIZE 1 - FOR I
    IF A I I SUB OBJ-> 2 MOD 1 == THEN
      1 'B' STO+
    END
  NEXT
  B
>>
```

The answers I have now are

1. 4175734221719/556712812325 approx 3 / 4, 0.75
2. 614074 / 1615189 very approx 2 / 5, 0.4
3. 1 /1
4. 1089 / 1629256, 0.000668 approx 8 / 30000
5. 10 / 99
6. 91720561 / 343614632 = 2.66929133856
7. 3.14159265357 slightly better approximation of PI than last time
8. 200100000001 / 1000000 -> 200100

Any better?

Chris

Re: Your score up to now

*Message #15 Posted by **Valentin Albillo** on 2 Feb 2006, 6:54 a.m.,*
in response to message #14 by Chris Dean

Hi again, Chris:

Chris posted: *"I have made a few changes to my code and have had another go."*

> I'm truly glad you did and thank you for your continued interest. In view of the scarce number of entries this challenge is getting, I think maybe it's useful to separate mice from men, speaking from a programming-abilities point of view, of course; though I don't get why, it's actually very easy, the novelty lies in the highly unexpected results.

"The answers I have now are 1. 4175734221719/556712812325 approx 3 / 4, 0.75"

> Nope, wrong value.

"2. 614074 / 1615189 very approx 2 / 5, 0.4"

> Wrong value.

"3. 1 /1"

> Wrong value.

"4. 1089 / 1629256, 0.000668 approx 8 / 30000"

> Wrong value.

"5. 10 / 99"

> Correct fraction, thus correct value. Do you think this is the *exact* value of this sum ? Or else, how good an approximation ?

"6. 91720561 / 343614632 = 2.66929133856"

> Correct value within a few ulps, wrong fraction. Once again, remember the 2/3 versus 166666666667/250000000000 I mentioned in the preamble to the challenge, when trying to convert the approximate result 0.666666666668 to fractional form. The long fraction is of course the minimal fraction which will result in a value of 0.666666666668, but this is misleading, since that final "8" is just an artifact of finite precision. Your

conversion to fraction must take these small inaccuracies into account and come instead with the simplest fraction that represents your result within a few ulps. That way you'll get the correct 2/3 instead of the artificious 166666666667/250000000000.

"7. 3.14159265357 slightly better approximation of PI than last time"

Correct value. Do you think that Pi is the *exact* value of this sum ? Or else, how good an approximation ?

"8. 200100000001 / 1000000 -> 200100"

Correct value, wrong fraction. The same comment as for #6 above applies. If you got S = 200100.000001, your conversion to fraction must consider, taking into account small inaccuracies, the "integer" fraction 200100/1 = 200100 is the lowest possible "fraction", not 200100000001/1000000

And as always, would you say that your fraction (or integer value) is the *exact* value of this infinite sum ? And if not, just how close is it ?

Well, there's been some improvement. But you really need to work your functions definitions and/or your summation method for the first 4 sums because your answers are not even close to the true results. The conversion from (approximate) computed sum to lowest fraction also needs reworking.

Thanks again for your continued efforts, and

Best regards from V.

Re: Your score up to now

*Message #16 Posted by **Chris Dean** on 2 Feb 2006, 7:45 a.m.,*
in response to message #15 by Valentin Albillo

```
Valentin


Thank you for your comments I will re-try 1 to 4. J-F has noticed an error in my program which is possibly the problem with my results. Also antoher problem I am possibly having is with the
With regards to my fraction values I am using the ->Q function which converts decimal values to the results as a fraction.


From your comments 1 to 4 I will retry.


For 5 this is the result output from the HG49G+ and I assume is exact (although I have a suspicion that 1 /10 might be the correct solution!)


For 6 I have 917207561 / 343614632, using a common factor of 2705627 gives me the fraction 339 / 127


For 7 possibly the accuracy is restricted to the accuracy of the HP49G+ but I would have to investigate further.


For 8 Point taken I used 200100 / 1 the first time round!


Thanks for the comments


Chris
```

Edited: 2 Feb 2006, 7:52 a.m.

Re: Your score up to now

*Message #17 Posted by **Arnaud Amiel** on 2 Feb 2006, 10:53 a.m.,*
in response to message #16 by Chris Dean

Using ->QPI sometimes provides "better" fractions.

I will have to try more that (3) and recheck it. Hopefully this weekend

Arnaud

S&SMC #13: Adding up to infinity

*Message #18 Posted by **Valentin Albillo** on 2 Feb 2006, 11:17 a.m.,*
in response to message #17 by Arnaud Amiel

Please do. And thanks a lot for your interest :-)

(By the way, both your value and fraction for #3 *are* correct. Would you state whether you deem the fraction to be the *exact* value for the infinite sum ? Perhaps computing some more extra digits would allow you to decide ? (hint, hint) ;-)

Have a busy and enjoyable weekend, keep up the good work, and

Best regards from V.

Re: S&SMC #13: Adding up to infinity

*Message #19 Posted by **Arnaud Amiel** on 2 Feb 2006, 5:46 p.m.,*
in response to message #18 by Valentin Albillo

A bit more work has been done. I will now try to follow your hint. In the mean time I hope this will help others to improve:

I have had a quick look and decided to use my 49G+.

The advantage of the 49G+ is that it is fast so I could use very inefficient algorithms.

I have up to now only looked at those not involving individual digits. I plan to do those this week end in ML as I believe it is the only way to properly deal with individual digits; and it is fun... so more results a bit later.

Anyway, here I go.

I use a program F to define f(N)
I use a program LOOP to loop infinitely (this is where I use the speed of the 49G+)
I use a LAUNCHER program to catch the error when LOOP underflows
I use the ->QPI function to give me the fraction as I found it usually works better than ->Q for approximate results

```
LAUNCHER
<< IFERR LOOP THEN DEPTH 1 - DROPN END >>
```

This works from an empty stack because LOOP always leave the sum on the top level.

For S3:

```
LOOP
<< 0 0 WHILE 1 REPEAT 1 + 2 OVER ^ DUP F SWAP / ROT + SWAP END >>


F
<< XPON 2 MOD >>
```

I find 0.111219512196 the 49 tells me it is 114/1025 I would however believe it would be 114/1024 because we are looking at sums of 2^N

For S6:

```
LOOP
<< 0 0 WHILE 1 REPEAT 1 + 2 OVER DUP F UNROT ^ / ROT + SWAP END >>


F
<< PI 2 / * IP >>
```

I find 2.66929133856 the 49 tells me it is 339/127 I would however believe it would be 342/128=171/64 because we are looking at sums of 2^N

For S7:

```
LOOP
<< 0 0 WHILE 1 REPEAT 1 + DUP F ROT + SWAP END >>


F
<< SQ neg 100 / EXP >>
```

After dividing by 5, adding .1 and squaring I get 3.14159265357 which the 49 tell me is PI, and which I somehow believe

For S8:

```
LOOP
<< 0 0 WHILE 1 REPEAT 1 + 5 OVER DUP F UNROT ^ / ROT + SWAP END >>


F
<< PI 163 SQRT 3 * / EXP * IP >>
```

I find .31250000512 the 49 tells me it is 3814696/12207027 somehow I don't trust numbers after so many 0s so I would tend to say 5/16.

Anyway, I will try to find the others by using an F written in ML and will look again at calculating or intuiting exact results on the paper a bit better this weekend.

Arnaud

Edited: 3 Feb 2006, 2:18 a.m.

Re: S&SMC #13: Adding up to infinity

*Message #20 Posted by **Valentin Albillo** on 3 Feb 2006, 4:36 a.m.,*
in response to message #19 by Arnaud Amiel

Hi, Arnaud:

Arnaud posted:

"In the mean time I hope this will help others to improve"

> *THAT'S* exactly the idea behind these challenges o'mine: that everyone can enjoy then, and that everyone can learn something useful. Me, I'm enjoying a lot the ideas and code posted here and learning new RPL tricks and techniques all the time. I'm sure many others do too, even if they simply 'lurk' without posting a thing.
>
> Matter of fact, I firmly believe that for every forum visitor who posts something, there are at least ten others that do never post anything, but nevertheless do enjoy the topics under discussion.

"I find 0.111219512196 the 49 tells me it is 114/1025 I would however believe it would be 114/1024 because we are looking at sums of 2^N"

```
Yes, but you have:  computed Sum = 0.111219512196
                    and 114/1025 = 0.111219512195
                    but 114/1024 = 0.111328125000
```

> so it all depends on how much you do trust your computed Sum. If you're certain that 11-12 digits are correct ... Perhaps getting some more digits would help you decide. Are there any multiprecision libraries available for your calculator ? They would probably be useful here.

"I find 2.66929133856 the 49 tells me it is 339/127 I would however believe it would be 342/128=171/64 because we are looking at sums of 2^N"

Same argument here. It all depends on how much you trust your computed sum.

"For S8 [...] I find .31250000512 the 49 tells me it is 3814696/12207027 somehow I don't trust numbers after so many 0s so I would tend to say 5/16."

Double-check the sum definition and/or your code, because your result is way out.

"Anyway, I will try to find the others by using an F written in ML and will look again at calculating or intuiting exact results on the paper a bit better this weekend.

Great ! Have an enjoyable weekend and please, *do post* your ML code for others to see. It'll be most instructional to all of us, myself included. Thanks for your interest and

Best regards from V.

Re: S&SMC #13: Adding up to infinity

*Message #21 Posted by **Arnaud Amiel** on 3 Feb 2006, 3:59 p.m.,*
in response to message #20 by Valentin Albillo

> Quote:
>
> Are there any multiprecision libraries available for your calculator ? They would probably be useful here.

The 49 has the ability to deal with unlimited precision numbers which are built in. It is just a pitty that the functions themselves are not built in. There are however very good libraries to deal with those. I might try weather permitting.

> Quote:
>
> Double-check the sum definition and/or your code, because your result is way out.

Of course you are right, I miss read thedefinition of f(N) which should be << PI 3 / 163 SQRT * EXP * IP >> which gives me 200100.000001 which the hp tells me is 200100.

Now I have to work on the other ones

Arnaud

Re: S&SMC #13: Adding up to infinity

*Message #22 Posted by **Arnaud Amiel** on 4 Feb 2006, 5:45 p.m.,*
in response to message #20 by Valentin Albillo

I have had time to play around a little bit with ML. I jus coded the missing F. The other part is the same RPL

as above. I will try to comment enough for people not too familiar with Saturn ASM to understand.

Here we go:

For 1:

```
GOSBVL =POP1%  %We take real from the stack, save the context, put it in register A and set the processor in BCD


mode
ASL.W          %We shift A left to align the first digit with the S (4 bits) field of the register
C=0.W          %C will be our counter
P=4            %P is one nibble which we will increase until it carries. There are 12 digits in the mantissa of


A
{
SB=0           %The sticky bit is set when a shift operation loses a digit.
ASRB.S         %We shift the nibble in S by one bit to the right
?SB=0 { C+1.A }%If the sticky bit is still 0 we skip increasing the counter
```

```
ASL.W          %We shift A by one nibble to bring the next digit in the S field
P+1 UPNC       %We increment our counter P and go up to the first bracket if we haven't overflowed yet
}
%Pushing reals is a trouble so on a 49 we push a ZINT. Much easier
A=C.A P=3      %We copy our counter C in A which will be pushed - P holds the number of digits.
SETHEX         %We must always return in HEX mode not BCD
GOVLNG =PUSHzintLoop %We push the ZINT and return
@
```

This gives me .111111111108 which ->QPI tell me is 1/9 and which I believe to be exact.

For 2: First I believed it would be a straightforward copy of the above but it turned out not to be for two reasons.

1) There is no ?SB<>0 test
2) For small numbers the registers are full of zeros which are even digits

So after playing a bit I got this:

```
GOSBVL =POP1%  %Same as before
SETHEX         %We want to calculate how many digits we will look at to load P in HEX
B=0.A          %B will hold the result
B=A.P          %We get the LSD of the exponent of A stored in DCB
ASR.X          %We get the next digit in the P=0 field
?A=0.P { B+10.A } %If it is not 0 we add 10 - I only consider "small" numbers to < 1E13
LC 00F C-B.A P=C 0 %We set our counter initial value in C and move it to P
SETDEC         %We will count the digits in DCB
C=0.A          %As before
ASL.W
{
C+1.A          %As there is no ?SB<>0, we assume even
SB=0 ASRB.S ?SB=0 { C-1.A } %If we were wrong we decrement our counter
ASL.W          %The rest is as before
P+1 UPNC
}
A=C.A P=3
SETHEX
GOVLNG =PUSHzintLoop
@
```

This gives me 1.03160638597 which ->QPI tell me is 3166/3069. I just don't believe that Valentin would have us

looking for this fraction. So either my calculation is not accurate enough or there is something special about

this fraction that I don't know.

For 4 and 5:

By now I had gotten in the mood of writing ML, which I usually only do when Valenin posts his cahllenges. So I

figured out the best method for this one was different. Here we go.

```
GOSBVL =POP1%   %Same as before
LC 101010101010000 %We use a mask to get only the LSB of even placed digits
SB=0 ASRB.X ?SB=0 { CSL.M } %If the exponent is odd we shift the mask
A&C.W           %We AND everything
P=4             %We will go through the whole 12 digits
ASL.W           %We bring the first digit to the end of the register
{
ASLC            %We circularly shift the first digit to the beginning of the register
C+A.X           %Add it to our counter
A=0.X           %Clean the X field of the A register to avoid adding old digits
P+1 UPNC        %Same as before from here
}
A=C.X P=3
SETHEX
```

```
GOVLNG =PUSHzintLoop
@
```

This gives me 1.01010101008E-2 and .10101010101 which the calc tell me are 1/99 and 10/99 respectively which I

believe.

Now I still need to recheck a few of these sums to see if I can come up with something more reasonable. It is

time to recheck the manual for the extended precision libraries.

Hopefully a bit more information tomorrow.

Arnaud

Re: S&SMC #13: Adding up to infinity - Rechecked with more numbers

*Message #23 Posted by **Arnaud Amiel** on 5 Feb 2006, 7:43 a.m.,*
in response to message #22 by Arnaud Amiel

Up to now, I was happy that: S1=1/9
S4=1/99
S5=10/99
S7=PI
S8=200100

For the others, Valentin seemed to imply that getting more digits would help so I used the Longfloat library it is much slower but I got many more digits.

S3 was 0.1112195121951219512195..... so I am happy that S3 is 114/1025

S2 was 10316063859718040101220361779253392004437687402157981960542380195161838125925590758044055067926010 15.E-99 and however hard I try, I can not find a period in there.

S6 is 2669291338582677165354330708661417322834645669291338582677165354330699315386077008072098943659183497.E-99 and there seem to be a period but it is quite large 669291.... this yields a long fraction: 29658792650918635170603675406824146981627/11111111111111111111111111111111111111111 Somehow it looks nicebut I still can't say that I am happy with it.

So unless a miracle happens before the end of the weekend, I lose 2 points for S2 and S6

And I am still wondering how Valentin got all these sums?

Arnaud

Edited: 5 Feb 2006, 7:45 a.m.

Re: S&SMC #13: Adding up to infinity - Rechecked with more numbers

*Message #24 Posted by **PeterP** on 5 Feb 2006, 2:11 p.m.,*
in response to message #23 by Arnaud Amiel

Nice work Arnaud! Seems that between you and J-F, the French own this challenge. I was hoping to learn again something magnificent from Markus or Ernst but they were notably abscence. So were all the other forum-prowling code magicians who write in RPN, I always hope to take a peek and learn some new trick from them...

And - at least from my low vintage point - Valentin is anyway in a different sphere...

However, I hope that he creates some challenges that are not only doable by my little-old-41 (albeit it carres a y in its name) yet also at least carry the change to win something for it as well, rather than those fabulous programs for the 71, which I do not have... How about a Sudoku solver for the 41, Valentin?

Can't wait to have the magician lift the curtain and tell the story about those sums. I bet there is some unifying theme between them, too many 1/9, 1/99, 10/99...

Cheers

Peter

Better than ->Q or ->Qpi

*Message #25 Posted by **Gene** on 2 Feb 2006, 11:29 a.m.,*
in response to message #17 by Arnaud Amiel

would be joe horn's amazing dec->frac routine, if you have that available. I think he's posted the routine to comp.sys.hp48 in the past.

Re: Better than ->Q or ->Qpi

*Message #26 Posted by **Arnaud Amiel** on 3 Feb 2006, 2:32 a.m.,*
in response to message #25 by Gene

I believe you are talking about PDQ. I was thinking about using it but decided to restrict myself to built in functions. I will give it a try if I have time this weekend.

Arnaud

Edited: 3 Feb 2006, 2:34 a.m.

Re: Your score up to now

*Message #27 Posted by **J-F Garnier** on 2 Feb 2006, 7:05 a.m.,*
in response to message #14 by Chris Dean

Chris,

I think you are calculating f(N^2) instead of f(2^N). I find totally different results on my HP71.

J-F

Re: Your score up to now

*Message #28 Posted by **Valentin Albillo** on 2 Feb 2006, 7:08 a.m.,*
in response to message #27 by J-F Garnier

Jean-François wrote:

"I find totally different results on my HP71"

> And why on earth don't you post them, man !? :-)

Best regards from V.

Re: Your score up to now

*Message #29 Posted by **J-F Garnier** on 2 Feb 2006, 8:11 a.m.,*
in response to message #28 by Valentin Albillo

And why on earth don't you post them, man !? :-)

Ok, ok ... The programming task is quite simple, well maybe not the decimal->fraction conversion, but as we are using "high level" functions like ->Q or FRAC$...

No, the fact is that I'm unable to judge the results, especially for question #1 to #5. Personally, I prefer the questions #6 to #8.

About #7, here is my result and comment:

```
90 DEF FNF7(N)=EXP(-N*N/K)
100 S=0
105 K=100
110 FOR N=100 TO 1 STEP -1
120 S=S+FNF7(N)
130 NEXT N
135 S=(S*2+1)^2/K
140 DISP S
```

Result is: 3.14159265357 close to pi, as already posted.

Note that I rewrote the sum as

```
S = ( (SUM( N=1, N-> Inf, f(N) ) * 2 + 1 ) ^2 ) / K
where f(N) = e^(-N*N/K)   and K=100 in Valentin's problem
```

I get the same result (up to 11 digits) for K value from 100 down to 3. But for K=2, I get 3.1415926872. So my guess is that the sums are very good approximations of PI (but are not PI), and the limit when K -> Inf may be actually PI.

About question #8: with a similar program I found : 200100, also already posted. I have some doubt that the exact value is an integer, as it seems related to the famous exp(pi*sqrt(163)) number, which is close to an integer, but is not.

J-F

Edited: 2 Feb 2006, 11:08 a.m.

Re: Your score up to now

*Message #30 Posted by **J-F Garnier** on 2 Feb 2006, 12:57 p.m.,*
in response to message #28 by Valentin Albillo

Well, Valentin, here are my results for problems 1 to 5:

```
10  DEF FNF1(N)
20    C=0 @ P=1
30    WHILE N>=1
51      IF MOD(N,2)=1 THEN C=C+1 ! PB#1
52      ! IF MOD(N,2)=0 THEN C=C+1 ! PB#2
53      ! C=C+1 ! PB#3
54      ! IF MOD(N,2)=1 AND MOD(P,2)=1 THEN C=C+1 ! PB#4,5
60      N=N DIV 10
65      P=P+1
70    END WHILE
81    FNF1=C ! PB #1,2,4,5
83    ! IF MOD(C,2)=0 THEN FNF1=1 ELSE FNF1=0 ! PB#3
90  END DEF
100 S=0
110 FOR N=100 TO 1 STEP -1
120   S=S+FNF1(2^N)/2^N ! PB#1,2,3,4
125   ! S=S+FNF1(N)/10^N ! PB#5
130 NEXT N
140 DISP S;FRAC$(S)


Results:
#1: .11111111111 1/9
#2: 1.03160638645 3166/3069
#3:  .111219512195 114/1025
#4:  .010101010101 1/99
#5:  .10101010101 10/99
```

Results #1,4,5 are beautiful enough to be true, I don't know for #2 and #3...

Waiting for your final comments!

J-F

Re: Your score up to now

*Message #31 Posted by **Valentin Albillo** on 3 Feb 2006, 5:42 a.m.,*
in response to message #30 by J-F Garnier

Hi, Jean-François:

Jean-François posted: *"Results #1,4,5 are beautiful enough to be true, I don't know for #2 and #3 [...] Waiting for your final comments!"*

> Thanks a lot, you've done a great job. Your exact score, next Monday.
>
> In the meantime, I would never ask this of anyone else, but I know you're fully capable (you did once in a former challenge): *how about getting more precise results, say to 20 digits or more ?* That would probably help you refine your final judgment about the exactness of your results, right ? Or perhaps not ? :-)

Full answers and comments next Monday. Have a nice weekend, and

Best regards from V.

Re: Double precision [quite long]

*Message #32 Posted by **J-F Garnier** on 4 Feb 2006, 6:57 a.m.,*
in response to message #31 by Valentin Albillo

I had to take up this challenge!

```
Problems #1 to #5:
10  DEF FNF2(H,L)
20    C=0 @ P=1
30    WHILE H>0 OR L>0
40      IF MOD(L,2)=1 THEN C=C+1 ! PB#1
50      ! IF MOD(L,2)=0 THEN C=C+1 ! PB#2
60      ! C=C+1 ! PB#3
70      ! IF MOD(L,2)=1 AND MOD(P,2)=1 THEN C=C+1 ! PB#4,5
80      ! calculate [H,L] / 10
90      L=L DIV 10+MOD(H,10)/10*M0 @ H=H DIV 10
100     P=P+1
110   END WHILE
120   FNF2=C ! PB#1,2,4,5
130   ! IF MOD(C,2)=0 THEN FNF2=1 ELSE FNF2=0 ! PB#3
140 END DEF
150 S=0
160 H0=0 @ L0=2 ! integer number under test, start with 2.
170 M0=1.00000000000E+12 ! 1E12 limit between H0 and L0
180 H1=0 @ L1=0 ! real result
190 M1=.0000000001 ! 1E-10 limit between H1 and L1
200 FOR N=1 TO 100
210   X=FNF2(H0,L0)
211   ! X=FNF2(0,N) ! PB#5
220   ! calculate X/2^N in dble precison
230   H2=X @ L2=0
240   FOR J=1 TO N
250     H2=H2/2 @ L2=L2/2 ! PB#1,2,3,4
255     ! H2=H2/10 @ L2=L2/10 ! PB#5
260     X=MOD(H2,M1) @ H2=H2-X @ L2=L2+X @ IF L2>=M1 THEN L2=L2-M1 @ H2=H2+M1
270   NEXT J
280   ! add H2,L2 TO H1,L1
290   H1=H1+H2 @ L1=L1+L2 @ IF L1>=M1 THEN L1=L1-M1 @ H1=H1+M1
300   ! calculate [H0,L0] *2
310   H0=H0*2
320   IF L0>=M0/2 THEN L0=L0-M0/2 @ H0=H0+1
330   L0=L0*2
340   DISP H1;L1
345   IF H0>1.E+12 THEN EXIT N
350 NEXT N
```

```
Problem #6:
5   DEF FNF(N)=IP(PI/2*N)
150 S=0
180 H1=0 @ L1=0 ! real result
190 M1=.0000000001 ! 1E-10 limit between H1 and L1
200 FOR N=1 TO 100
210   X=FNF(N)
220   ! calculate X/2^N in dble precison
230   H2=X @ L2=0
240   FOR J=1 TO N
250     H2=H2/2 @ L2=L2/2
260     X=MOD(H2,M1) @ H2=H2-X @ L2=L2+X @ IF L2>=M1 THEN L2=L2-M1 @ H2=H2+M1
270   NEXT J
280   ! add H2,L2 TO H1,L1
290   H1=H1+H2 @ L1=L1+L2 @ IF L1>=M1 THEN L1=L1-M1 @ H1=H1+M1
340   DISP H1;L1
350 NEXT N


Results:
#1: .1111111111  1.11111111128E-11
  Last two digits are supposed to be invalid.
#2: 1.0316063864  4.50961225158E-11
  in line the 3166/3069 fraction, as can be checked with the HP71:
                                 1.0316063864450961225158
  3166/3069                   -> 1.031606386
  (3166-3069*1.03160638)/3069 ->          6.44509612252E-9
#3: .1112195121   9.51219512197E-11
  check:                       .1112195121951219512197
  114/1025                  -> .1112195
  (114-1025*0.111219)/1025 ->       5.12195121951E-7
#4: .0101010101  1.01010101013E-12
#5: .101010101  1.0101010101E-11
#6: 2.6692913385  8.26771653543E-11
  check:                    2.6692913385826771653543
  339/127                 -> 2.6692913
 (339-2.669291*127)/127  ->        3.38582677165E-7
```

With 20 digits, the fractions represent the sums accurately. Are they the exact values? I don't know. If I was a mathematician, I would write a 100-digit accuracy program (or use a PC Math software) and would consider spending some time on trying to prove it if the results still match. But I'm not a mathematician...

I was a little disapointed, because I expected that at least one result would invalidate the fraction approximation. Valentin insisted too much on getting our comitment on rational results! I'm really intrigued by problem #6, how a combination of the decimals of PI could make a rational number?

Valentin, we need you!

J-F

Re: Your score up to now

*Message #33 Posted by **Chris Dean** on 2 Feb 2006, 7:49 a.m.,*
in response to message #27 by J-F Garnier

J-F

You are right thanks for that. I might possibly get the correct answers now!

Regards

Chris

Re: Your score up to now

*Message #34 Posted by **Chris Dean** on 2 Feb 2006, 11:04 a.m.,*
in response to message #33 by Chris Dean

```
Valentin
Once again more results. You cannot fault me for trying!


1. 1 /9
2. 32 /31
3. 1 / 1. I am having problems with this one as I can also get the answer 0 if I change my program by adding an EVAL statement after 2^N.
4. 1 / 99
5. 10 / 99 This is the result output from the HG49G+ and I assume is exact.
6. I have 917207561 / 343614632, using a common factor of 2705627 gives me the fraction 339 / 127
7. 3.14159265357 possibly the accuracy is restricted to the accuracy of the HP49G+ but I would have to investigate further.
8. Point taken, I used 200100 / 1 the first time round!


Thanks for the comments


Chris
```

Edited: 2 Feb 2006, 11:18 a.m.

S&SMC #13: Adding up to infinity

*Message #35 Posted by **Valentin Albillo** on 2 Feb 2006, 11:34 a.m.,*
in response to message #34 by Chris Dean

Hi again, Chris:

Chris posted:

"You cannot fault me for trying!"

> Indeed !! :-)
>
> As for your new results:

- #1, #4, #5, #6 (the smaller fraction), #7 and #8 are Ok.
- #2 needs reworking: the fraction you give doesn't result in a value accurate enough.
- #3 is wrong.

Anyway, if you want to remain friends with me, you must abide by the challenge's regulations (see "Caveat emptor" in my original post), this is:

1. You *must* post your code. Posting just solutions won't do, and it's to be considered impolite. Share with your fellow forum visitors so that they can marvel at your code and learn how things are done !
2. For this particular challenge, please state your opinion on the exactness of the final results. Simply accepting some calculator's output without verification and jugdment is terribly bad practice. What would you do if I didn't comment on their correctness ? Would you trust anything important to these results ? You gotta be confident in what you get.

Enough ranting. Have a good weekend, I'll post my solutions next Monday, so you still have time to make it perfect.

Best regards from V.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #36 Posted by **PeterP** on 2 Feb 2006, 12:13 p.m.,*
in response to message #8 by Valentin Albillo

Okay valentin, here are some initial tries, just to see if I am on the right track. Unfortunately I do not own a 71, so a) my code is for a 41, which I hope is still okay.

The main loop is the "Lbl S", where one chooses which function F(N) to use. It right now is set up only for the first few, where we have to divide by 2^n.

Also, I have not written a conversion to fraction routine, dont have a good idea how to do this yet...

Hence I have only the fractions here:

```
1) 0.11111111, 1/9 (I believe exact)
2) 1.031606386
3) 0.11219512
```

here is the code, I will add the other functions piece by piece and add it here if there is any interest.

```
lbl 'SSMC13


"----Place some heavily used constants in regs for speed"
CLRG
50
E3/E+
Sto 00
Sto 60
2
Sto 61
48
Sto 62


"----Place 2^n in regs 01-50"
Lbl 15
  Rcl 61
  Rcl 00
  Int
  y^x
  Sto Ind 00
  Isg 00
Gto 15


Lbl 'S "---- S for Start
clx
Sto 51
'What F?'
prompt
'F
Aint
Asto 63
Rcl 60
Sto 00


Lbl 'L "---- L for Loop
  View 00
  Rcl Ind 00
  Xeq Ind 63
  Rcl Ind 00
  /
  Sto+ 51
  Isg 00
Gto 'L


View 51
Stop
Gto 'S


Lbl 'F1 "--- Sum of odd digits
Cla
Aint
Clx
Sto 52
Lbl 02
  Atox
```

```
  x=0?
  Gto E
  Rcl 62 "-- 48"
  -
  Rcl 61 "-- 2"
  Mod
  St+ 52
Gto 02



Lbl 'F2 "---> Sum of Even digits
Cla
Aint
Aleng
Sto 52
Lbl 01
  Atox
  x=0?
  Gto E
  Rcl 62 "-- 48"
  -
  Rcl 61 "-- 2"
  Mod
  St- 52
Gto 01


Lbl E
Rcl 52
Rtn



Lbl 'F3 "--- 1 if even number of digits
Cla
Aint
Sign
Aleng
Rcl 61 "--- 2"
Mod
-
Abs
Rtn
```

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #37 Posted by **PeterP** on 2 Feb 2006, 1:50 p.m.,*
in response to message #36 by PeterP

Some more functions and results, yet still no fractions...

```
4) 0.010101010, 1/99, believe is accurate
5) 0.101010101, 10/99, believe is accurate
6) 2.669291339
7) 3.141592654, Pi, believe is accurate
8) 200100, believe is accurate
```

I also made some small changes to the main program, to accomodate more constants etc, but nothing major (e.g. depending on the function I need to preload r01-r50 with different values (2^n for 1-4, 6; 1^n for 7, 10^n for 5 and 5^n for 8). The other functions f(N) are below

```
Lbl 'F4
Cla
Aint
Clx
Sto 52
Lbl 03
  E
```

```
  Chs
  Arot
  Atox
  X=0?
  Got E
  Rcl 62
  -
  Rcl 61
  Mod
  St+52
  RAdel
Gto 03


Lbl 'F5
Rcl 00
Int
Cla
Aint
Clx
Sto 52
Gto 03


Lbl 'F6
Rcl 00
Int
Pi
*
Rcl 61
/
Int
Rtn


Lbl 'F7
Rcl 00
Int
x^2
Chs
Rcl 64   "---> 10
x^2
/
e^x
Rtn


Lbl 'F8
Rcl 00
Int
Rcl 65  "---> exp(Pi/3*sqrt(163))
*
Rtn
```

The fraction part I will try another day. Enough playing for one day...

Cheers

Peter

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #38 Posted by **PeterP** on 2 Feb 2006, 6:27 p.m.,*
in response to message #37 by PeterP

here is the fraction part (could not stop my curiosity).

```
1) 98/95
2) 1/9
3) 1/9 (OR 114/1025 depending on Valentins desired accuracy)
4) 1/99
5) 10/99
6) 8/3 (OR 339/127 depending on Valentins desired accuracy)
7) Pi
8) 200100/1


Lbl 'D
'Prec
prompt
Sto 70
1 E-10
Sto 75
Rcl 51 'That's the value to be fractionalized
Sto 71
Sign
Sto 73
Clx
Sto 74


Lbl 10
 Rcl 75
 Rcl 71
 Enter
 Int
 -
 x<=y?
  Gto 'X "===> Exit


 1/x
 Sto 71
 Int
 Rcl 73
 *
 Rcl 74
 +
 x<>73
 x<>74
 Rcl 73
 Rcl 51
 *
 .5
 +
 Int
 Sto 72
 Rcl 73
 /
 Rcl 51
 -
 Abs
 Rcl 70
 x<=y?
  Gto 10


Lbl 'X
View 72
Stop
View 73
Stop
Rcl 72
Rcl 73
/
View x
Stop
```

```
Gto 'S
```

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #39 Posted by **Chris Dean** on 3 Feb 2006, 1:50 p.m.,*
in response to message #38 by PeterP

Valentin

I apologise for not including code in my previous post. I now have further results for problems 2 and 3 (including code!). For each case I have used the main program shown below

```
<<
  CLEAR
  'SUM(N=1,10000, F(N)/2^N)'    EVAL
  ->Q
>>
```

Where
F(N) defines the function for each problem,
SUM is the HP49G+ summation function
->Q is the HP49G+ quotient function which converts decimal results
as fractions.

2. F(N) is defined by

```
<<
  2 SWAP ^ DUP
   ->STR ‘A’ STO
  LOG 1 +  IP ‘B’ STO
  0 ‘C’ STO
  1 B FOR I
    IF A I I SUB OBJ-> 2 MOD 0 == THEN
      1 C = ‘C’ STO
    END
  NEXT
  C
>>
```

This gives a result of 3166 / 3069. I believe this to be accurate
for the algorithm I am using yielding the decimal result 1.03160638645.

3. F(N) is defined by

```
<<
  2 LOG * IP 2 MOD
>>
```

This gives a result of 114688217 / 1031187916 = 0.111219512196. I
believe this to be accurate for the decimal result that is
produced. The algorithm is dominated in this case by the value of
2^-N which is having little affect on the result as N increases.
Running the algorithm over 100 and 10000 iterations yielded
the same results as those for 39 iterations. Considering the 44th
iteration which is the next even result to be considered after the
39th, this result will add 5.68434188609E-14 to the previous sum.
The value does not have any affect on the current sum
0.111219512196 on my calculator. Looking at, say, 114 as the
denominator multiply by the result yields 1024.9999999999. It
might 'safe' to assume that a reasonable estimate for the result

could be 114 / 1025 which gives 0.111219512195. A close result to
the previously calculated one but is it a valid?

Regards
Chris

Edited: 3 Feb 2006, 2:27 p.m.

Re: Short & Sweet Math Challenge #13: Adding up to infinity

*Message #40 Posted by **werner** on 3 Feb 2006, 5:40 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin!
Well I want the prize, so I had better participate (and hope I'll pass the exam ;-)
I used the 49, for two reasons:
- Qpi to convert the answer to fractions
- long integer support allows me to run tests twice to see if the infinite sum really converges to the proposed fraction
for that I use the following program that (where possible) adds up the *exact* terms of the sum.

```
In: 2: f
    1: N
Out: 3: SUM(i=1,N,f(i)) (numeric)
     2: approximate fraction
     1: difference between exact and approximate fraction
@ INFS
\<<
  \-> F N
  \<<
    0
    1 N
    FOR I
      I F EVAL +
    NEXT
  \>>
  EVAL
  DUP \->NUM
  DUP \->Q\pi
  ROT OVER - EVAL \->NUM
\>>
@ F1 = odd(2^N)/2^N
\<<
  2 SWAP ^
  0 OVER \->STR
  DO
    DUP TAIL SWAP HEAD OBJ\-> 2 MOD ROT + SWAP
  UNTIL DUP SIZE NOT
  END
  DROP SWAP /
\>>
@ F2 = even(2^N)/2^N
\<<
  2 SWAP ^
  0 OVER \->STR
  DO
    DUP TAIL SWAP HEAD OBJ\-> 2 MOD NOT ROT + SWAP
  UNTIL DUP SIZE NOT
  END
  DROP SWAP /
\>>
@ F3 = evendigits(2^N)/2^N
\<<
  2 SWAP ^ DUP LENGTH 2 MOD NOT
  SWAP /
\>>
@ F4 = oddinoddplaces(2^N)/2^N
\<<
  2 SWAP ^
  0 OVER \->STR "0" SWAP +
  DO
```

```
    DUPDUP SIZE DUP SUB OBJ\-> 2 MOD ROT + SWAP
    1 OVER SIZE 2 - SUB
  UNTIL DUP SIZE NOT
  END
  DROP SWAP /
\>>
@ F5 = oddinoddplaces(N)/10^N
\<<
  0 OVER \->STR "0" SWAP +
  DO
    DUPDUP SIZE DUP SUB OBJ\-> 2 MOD ROT + SWAP
    1 OVER SIZE 2 - SUB
  UNTIL DUP SIZE NOT
  END
  DROP 10 ROT ^ /
\>>
@ F6 = Integer part of (Pi/2 * N)/2^N
\<<
  \pi 2. / OVER * IP R->I 2 ROT ^ /
\>>
@ F7 EXP(-N*N/100)
\<< SQ -100. / EXP \>>
eval with \<< 0. 50. 1. FOR I I F7 + NEXT 5 / 0.1 + SQ \>>
@ F8
\<< 163. \v/ 3. / \pi \->NUM * EXP OVER * IP R\->I 5 ROT ^ / \>>
results
                              50     100    200    EXACT?
1. 0.111111111111 '1/9'       7e-15  1e-29  2e-59    Y
2. 1.03160638644  '3166/3069' 7e-15  1e-29  1e-32    N
3. 0.111219512196 '114/1025'  8e-16  9e-32  1e-31    N
4. 0.010101010101 '1/99'      3e-15  5e-30  8e-60    Y
5  0.101010101010 '10/99'     1e-51 9e-101 1e-100    N
6. 2.66929133858  '339/127'   7e-14  1e-28  2e-58    Y
7. 3.14159265357   PI                                 ?
8. 200100.         200100                            N


7: no idea. I guess not
8: 200100 is NOT the exact value of the sum.  It would be if exp(pi/3*sqrt(163)) were an integer (640320) but it isn't.
If it were, the infinite sum could be written as
640320*(1/5 + 2/25 + 3/25 + .. N/5^N) = 640320*S
when T = 1/5 + 1/25 + 1/125 then 5*T = 1+T, so T=1/4
Then S = T + T/5 + T/25 + ... = T*(1+T) = 5/16
so 640320*5/16 = 200100
Hope I'll pass!
```

S&SMC #13: My Solutions & Comments [LONG]

*Message #41 Posted by **Valentin Albillo** on 5 Feb 2006, 3:07 p.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

First of all, I'm **absolutely** impressed with the quality & quantity of your posted solutions to this humble challenge of mine, it's been most enjoyable and unexpected.

Frankly, I thought for a while whether I should submit this particular challenge or some other easier one instead, because I feared people would consider it too difficult and too 'busy', what with eight different cases to solve. But then, on the other hand, the subject matter seemed to me irresistibly interesting, because these sums are truly weird, and the results are well-nigh *surreal*, so to say.

Why ? Well, most people knowledgeable in things mathematical have seen infinite sums (Taylor expansions, say) where each term is defined as some combination of elementary functions, such as trigonometrics, exponentials, or logarithms, mixed up with powers and arithmetic operations. Even other non-elementary functions are sometimes included as well, such as Gamma or Bessel functions, or some integrals, things like that.

But what is one to do with functions such as the one featured in Sum #1, where f(n) *counts* the number of odd digits in its argument ? How can we deal with such a function by analytical means ? Does it have a derivative ? Can it be integrated ? Can it be extended to arbitrary, non-integer arguments, much like the factorial function can ?

Such weird functions, which deal more with their argument's *form* than with their argument's *value*, so to say, seem absolutely *intractable*, and there's no telling what an infinite sum of them could amount to. Math intuition would seem to suggest that it would be some weird no-name irrational value, not related to any well-known mathematical constants. *Yet the first sum comes to 1/9 !* A simple rational !! Exactly !!!

And our amazement can only increase when the second sum, which features the twin function who counts the number of *even* digits in its argument, also seems to evaluate to a rational argument, though this time a not-so-simple fraction, namely 3166/3069.

At the very least, 10-digit and 12-digit results exactly agree with this fraction. Jean-François Garnier took my hint and came up with a 20-digit sum still in perfect agreement with the fraction. Yet, the sum does *not* exactly evaluate to this fraction but comes *incredibly close*. Matter of fact, the sum isn't neither rational nor even irrational, but *transcendental*. An easily-defined, easily-computable transcendental number which is extremely close to the rational 3166/3069. Why this asymmetry between the odd case (sum = 1/9, rational, exactly) and the even case (sum = 3166/3069, extremely close approximation but actually transcendental) ?

Fear not: there's a deep mathematical reason for these unexpected evaluations, these weird counting functions *can* be treated analitically, and sums #6, #7, and #8 also have a suitable and easily understandable explanation, only this is not the place to explain it in detail, of course. Let's go instead for the solutions proper, plus assorted comments.

My Solutions

As usual, I'll give my solutions for the HP-71B, as they're easier to understand and to translate to the programming language of any other HP models, say RPN or RPL.

First of all, we do need some way of converting decimals to fractions. Jean-François Garnier used the FRAC$ function available in the HP-71B's JPC ROM, but that's hardly portable to most other HP machines so here's a 5-line **DEC2FRC** (Decimal-to-fraction) subprogram I wrote for the occasion which will do the conversion:

```
100 SUB DEC2FRC(X,N,D,W) @ IF X=0 THEN N=0 @ D=1 @ END
110 U=0 @ V=1 @ N=1 @ D=0 @ Y=INF @ Z=ABS(X) @ F=SGN(X) @ X=Z @ W=ABS(W)
120 C=INT(X) @ IF FP(X)=0 THEN N=N*F @ END ELSE X=1/FP(X) @ S=N @ T=D
130 N=N*C+U @ U=S @ D=D*C+V @ V=T @ R=N/D @ IF ABS(R/Z-1)<W THEN N=N*F @ END
140 IF R=Y OR MAX(N,D)>1.E+12 THEN N=U*F @ D=V @ END ELSE Y=R @ GOTO 120
```

This subprogram is based on the continued fraction expansion of the given decimal number, and it simply computes the subsequent approximants till the tolerance is met, returning the last one. It works for most any argument within range, regardless of its sign or value.

To use it, you simply call DEC2FRC, passing it the following parameters:

```
 X: passed by value,     is the decimal number to convert to fractional form
 N: passed by reference, is the variable where the numerator will be returned
 D: passed by reference, is the variable where the denominator will be returned
 W: passed by value,     is a tolerance value which lets you control the
                         relative error of the fraction so you can get smaller
                         fractions which still result in acceptable errors;
                         specifying this tolerance as 0 will return the most
                         precise fraction (smallest error) found
```

For example, let's convert -PI to fractional form calling DEC2FRC from the command line:

```
>CALL DEC2FRC(-PI,N,D,0) @ N,D,N/D,PI

   -1146408    364913   -3.14159265359    3.14159265359
```

so our fraction is -1146408/364913, which agrees with PI to 12 digits. Let's suppose that we don't want so close an approximation, but prefer instead a simpler fraction. We'll call DEC2FRC again, but this time we'll specify a tolerance of 0.00001 for the maximum relative error:

```
>CALL DEC2FRC(-PI,N,D,1E-5) @ N,D,N/D,PI

     -355        113      -3.14159292035    3.14159265359
```

and so this time our fraction is -355/113, which agrees with -PI to nearly 8 digits, far better than our tolerance would have us believe. So our subprogram does work, and you can freely use it in your own programs which require fraction output. As for the sums themselves:

Sum #1:

S = SUM(N = 1, N -> Inf, f(2^n)/2^n)

where f(N) counts the number of *odd* digits (i.e.: 1,3,5,7,9) in N

Our main program will be:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 12/LGT(2) @ X=2^N
20 S=S+FNF(X,"13579")/X @ NEXT N @ CALL DEC2FRC((S),N,D,.000000001)
30 DISP USING "K,Z.10D,K,K,A,K";"Sum = ";S," = ",N,"/";D @ END
```

where FNF is a user-defined function that accept two parameters, the number whose digits will be counted, and a string argument specifying the digits to count. As we want to count the number of odd digits in N, we supply "13579" as the second parameter. Thus, FNF is defined so:

```
50 DEF FNF(X,D$) @ S$=STR$(X) @ C=0 @ FOR I=1 TO LEN(S$)
60 C=C+(POS(D$,S$[I,I])#0) @ NEXT I @ FNF=C @ END DEF
```

which you can test from the keyboard, if desired. This very same function will be re-used for Sum #2. Notice that the main program sums from N=1 to N=12/LGT(2). This is so because we don't want 2^N to be more than 12 digits in length, which is the maximum size for an exact integer argument in the 12-digit HP-71B's BASIC. Else, the count function would return *wrong* results.

Upon running the program we get:

```
>RUN


   Sum = 0.1111111111 = 1/9
```

This is indeed the *exact* result, a nice rational sum.

Sum #2:

S = SUM(N = 1, N -> Inf, f(2^n)/2^n)

where f(N) counts the number of *even* digits (i.e.: 0,2,4,6,8) in N

Our program and user-defined function are exactly the same, except in line 20 the string "13579" should be replaced by "02468".

Upon running it we get:

```
>RUN


   Sum = 1.0316063865 = 3166/3069
```

This result is only *approximate*, but *correct to 31 decimal digits* (!!)

Actually, we have:

```
   Exact value =  1.0316063864450961225154773541 87 130310+
     3166/3069 =  1.0316063864450961225154773541 87 031606+
```

and you can see that the true sum agrees with our fraction 3166/3069 to 31 decimal digits, but differs afterwards. Thus, the true sum for the even case is a *transcendental* number, but extremely close to a simple rational. This is in contrast with the sum for the odd case, which is exactly rational.

Sum #3:

S = SUM(N = 1, N -> Inf, f(2^n)/2^n) = 114/1025

where f(N) = 1 is N has an even number of digits, 0 otherwise

Our main program is extremely similar to the one for #1 and #2, and the user-defined function is much simpler:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 12/LGT(2) @ X=2^N
20 S=S+FNF(X)/X @ NEXT N @ CALL DEC2FRC((S),N,D,.000000001)
30 DISP USING "K,Z.10D,K,K,A,K";"Sum = ";S," = ",N,"/";D @ END
```

```
40 !
50 DEF FNF(X)=1-MOD(LEN(STR$(X)),2)
```

Upon running it, we get:

```
>RUN


   Sum = 0.1112195122 = 114/1025
```

Once again, this result is only approximate, but *correct to 30 digits* !

In fact, we have:

```
Exact value =  0.11 12195 12195 12195 12195 12195 12204 97+
   114/1025 =  0.11 12195 12195 12195 12195 12195 12195 12+
```

and so the exact sum, which begins mimicking a 5-digit period, actually ceases to agree with the nice fraction after 30 digits. The sum is also transcendental and, again, extremely close to a rational value. Computing to, say, 20 or 25 digit precision would led anyone to think that the 12195 period continued indefinitely, it's hard to believe that, *suddenly*, it *vanishes forever* upon reaching the 30th digit !

Sum #4:

S = SUM(N = 1, N -> Inf, f(2^n)/2^n) = 1/99 (exact)

where f(N) counts the number of *odd* digits in *odd* places in the decimal expansion of N.

In this case, our main program is exactly the same as #1, and the user-defined function is a simple variation of the one in #1:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 12/LGT(2) @ X=2^N
20 S=S+FNF(X,"13579")/X @ NEXT N @ CALL DEC2FRC((S),N,D,.000000001)
30 DISP USING "K,Z.10D,K,K,A,K";"Sum = ";S," = ",N,"/";D @ END
40 !
50 DEF FNF(X,D$) @ S$=STR$(X) @ C=0 @ FOR I=1 TO LEN(S$) @ J=LEN(S$)+I-1
60 C=C+(POS(D$,S$[I,I]) AND MOD(J,2)) @ NEXT I @ FNF=C @ END DEF
```

Upon running it we get:

```
>RUN


   Sum = 0.0101010101 = 1/99
```

which is the *exact* result. This time the sum is, as in #1, exactly rational.

Sum #5:

S = SUM(N = 1, N -> Inf, f(n)/10^n)

where f(x) is the same as in Sum #4.

We only need to change these lines in our main program for #4:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 12/LGT(2)
20 S=S+FNF(N,"13579")/10^N @ NEXT N @ CALL DEC2FRC((S),N,D,.000000001)
```

Upon running, it returns:

```
>RUN


   Sum = 0.1010101010 = 10/99
```

where most unexpectedly, as the function *is the very same* and the sum is very similar, the result is only approximate, but *correct to 99 digits* !!! The true results is, again, transcendental, but this time much closer to a rational value.

Why Sum #4 results in an exact rational (1/99) while the extremely similar Sum #5 results in a transcendental which agrees to 99 digits with 10/99 may seem as mysterious as the odd-even dissimilar results in Sums #1 and #2. Of course, the "10" period dissappears upon reaching the 100th digit, but anyone computing the sum to 10, 20, 30, ..., 70, 80, 90 decimals would be adamantly convinced that it would go on forever.

Sum #6:

S = SUM(N = 1, N -> Inf, f(n)/2^n)

where f(x) = Integer part of (Pi/2 * x)

The main program is a trivial variation, and the user-defined function is also trivial:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 12/LGT(2)
20 S=S+FNF(N)/2^N @ NEXT N @ CALL DEC2FRC((S),N,D,.000000001)
30 DISP USING "K,Z.10D,K,K,A,K";"Sum = ";S," = ",N,"/";D @ END
40 !
50 DEF FNF(X)=INT(PI/2*X)
```

Upon running it we get:

```
>RUN


    Sum = 2.6692913385 = 339/127
```

and, again, this result is only an approximation to the real sum, which is transcendental. However, it's a very close approximation once more, this time *the fraction agrees with the true sum to almost 69 digits* !

```
Exact value = 2.669291338582677165354330708661417322834645669291338582677165354330 69931+
    339/127 = 2.669291338582677165354330708661417322834645669291338582677165354330 70866+
```

Sum #7:

S = ((SUM(N = 1, N -> Inf, e^(-n*n/100))/5+0.1)^2

The main program is very simple, no need to call DEC2FRC, and the user-defined function is trivial, too

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 50
20 S=S+FNF(N) @ NEXT N @ S=(S/5+.1)^2
30 DISP "S = ";S @ END
40 DEF FNF(X)=EXP(-X*X/100)
```

Unpon running it, we get

```
>RUN


   S = 3.14159265357
```

which makes us wonder if the exact sum would indeed be Pi. But lo and behold, *it isn't Pi* but another transcendental number which nevertheless *agrees with Pi to 428 decimal places* !! As my 13-year old daughter Laura put it: *"It looks like Pi, it smells like Pi, it tastes like Pi, but it isn't Pi"*

```
Exact value of the sum =


 3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186146740+


while Pi =


 3.141592653589793238462643383279502884197169399375105820974944592307816406286208
```

```
899862803482534211706798214808651328230664709384460955058223172535940812848111 74
502841027019385211055596446229489549303819644288109756659334461284756482337867 83
165271201909145648566923460348610454326648213393607260249141273724587006606315 58
817488152092096282925409171536436789259036001133053054882046652138414695194151 16
094330572703657595919530921861 17381+
```

Sum 8:

S = SUM(N = 1, N -> Inf, INT(N*e^(Pi*Sqrt(163/9)))/ 5^N)

Again, an extremely simple main program and straightforward user-defined function:

```
10 DESTROY ALL @ STD @ S=0 @ FOR N=1 TO 18
20 S=S+FNF(N)/5^N @ NEXT N
30 DISP "S = ";S @ END
40 DEF FNF(X)=INT(X*EXP(PI*SQRT(163/9)))
```

Upon running it, we get:

```
>RUN


    S =  200100
```

and, as you may suspect by now, this neat integer value is too good to be true and it's only an *approximation* to the exact value of the infinite sum which is *transcendental*. However, there's no typing both values for you to see the difference, *because 200100 agrees with the exact sum to at least 500,000,000,000 (half a billion) decimal digits* !!!!

So it's doubtful you could use your arbitrary precision math package to empirically test this claim. Now, having such a simple sum return a result which *differs from an integer by less than 10^500000000000* is simply mind-boggling !

The Prizes

Everyone who posted their solutions and estimates did extremely well, with special mentions to Arnaud Amiel (who coded some of the functions in Saturn assembly language and later tried multiprecision), Jean-François Garnier (who not only got all solutions Ok but coded special double precision programs to compute results to double accuracy), Werner (who got all results Ok, plus gave some math insight as to why some of them couldn't be exact), Chris (who kept trying once and again till he got the answers), and PeterP (who did his best with his trusty 41C).

All of you, either explicitly mentioned above or not, will receive by e-mail whatever PDF articles you want me to send you, please contact me either via this web site (clicking on my name at the beginning of each post of mine) or else via my e-mail address which you can find at my web page. Please send a valid e-mail address plus a list of the articles you do want, and I'll send them to you at once.

Thanks to all of you posters and lurkers for your interest, and

Best regards from V.

Re: S&SMC #13: My Solutions & Comments [LONG]

*Message #42 Posted by **Chris Bennett** on 5 Feb 2006, 5:05 p.m.,*
in response to message #41 by Valentin Albillo

Valentin,

Although I didn't participate, I am once again blown away by this "humble challenge" of yours, and left absolutely breathless (stunned, even) by the mathematical elegance of the results.

Valentin wrote:

> Quote:
>
> ...there's a deep mathematical reason for these unexpected evaluations, these weird counting functions can be treated analitically, and sums #6, #7, and #8 also have a suitable and easily understandable explanation, only this is not the place to explain it in detail, of course.

I'd love to learn the mathematical reasoning you allude to. If not here, could you at least point me toward a reference that explains this numerical sorcery?

Chris

Re: S&SMC #13: My Solutions & Comments [LONG]

*Message #43 Posted by **PeterP** on 5 Feb 2006, 10:05 p.m.,*
in response to message #41 by Valentin Albillo

Valentin,

Thanks again for your wonderful challenge - seems my 41 did much better than me, getting all the results right yet my intuition being wrong... (about both which ones are accurate and what precision you would want the fractions to be. Was quite happy with my Dec2Frac program though, fast and using a similar method as yours)

Same as Chris I'd love to learn more about the analytical/mathematical way of handling those sums you alluded to in your post.

As for your articles, - I'd love to read your two Sudoku solvers, maybe I can find a way to translate them to the 41. Also the Fantastic Four would be great - still working on the last couple of SSMC's, one of which is the high precision arithmetic package for the 41, as well as the Mean Matrices, as this is something I often have to tackle at work. Would love to learn your insights here. If you'd be so kind and email it to me (click on my name), that'll be wonderful.

As usually you have inspired me (and us I'd say) a lot over the last few days. So my next question should be no surprise to you:

When is the next SSMC coming?

Cheers

Peter

Re: S&SMC #13: My Solutions & Comments [LONG]

*Message #44 Posted by **Valentin Albillo** on 6 Feb 2006, 4:44 a.m.,*
in response to message #43 by PeterP

Hi, PeterP:

PeterP wrote:

"Thanks again for your wonderful challenge - seems my 41 did much better than me, getting all the results right yet my intuition being wrong... (about both which ones are accurate and what precision you would want the fractions to be. Was quite happy with my Dec2Frac program though, fast and using a similar method as yours)"

> You're welcome, and it's always nice to see a 41C being used for these challenges. That machine's still got a lot of life on it.

"Same as Chris I'd love to learn more about the analytical/mathematical way of handling those sums you alluded to in your post [...] As for your articles, - I'd love to read your two Sudoku solvers [...] Also the Fantastic Four would be great [...] as well as the Mean Matrices [...] if you be so kind and email it to me (click on my name), that'll be wonderful.

> Of course, they'll be on their way at once.

"As usually you have inspired me (and us I'd say) a lot over the last few days. So my next question should be no surprise to you: When is the next SSMC coming?"

> Thanks for your kind comments and appreciation, it does wonders for me to justify the time and effort it all takes. As for the next challenge, I try not to overdo it, so I usually wait a month or so between challenges. Expect #14 by March, 5th or so.

Best regards from V.

[Return to Index | Top of Index]

Go back to the main exhibit hall