# 2G03 Homework 4, 2020

Due: Suggested Tuesday, Oct 20th (Hard deadline Thursday Oct 22nd)
The files are available on phys-ugrad in `/home/2G03/HW4`
**Please follow the guidelines on code and homeworks. You will use a directory called *HW4* in your area for these programs and Makefiles for them. Each program should have a subdirectory with its name. When you copy the files provided, these directories will be automatically created. The grader will look there to locate and run your programs.**

**Some code and Makefiles are already provided. You can copy them all to your area (which also creates any all directories and files within them) with:**
`cp -r /home/2G03/HW4 ~/`
`HW4` **already contains code. You should start with the C++ files there and change them rather than creating new files yourself.**

**For full credit you must provide C++ code for exercise marked with** `PROGRAM` **on** `phys-ugrad` **and responses to ALL the questions below. Make sure there is a final working version of the program source file** `pi.cpp` **on** `phys-ugrad` **in your HW4 directory. You do not need to print out the Makefile or code. Responses to questions can be hand-written and scanned but typed in a file is preferred.**

## Numbers and Functions

### Exercise `PROGRAM` *Estimating Pi*

When a non-trivial math function like cosine, log or square root is implemented in computer hardware or software it is often in the form of the sum of a series. The trick is to come up with a series that rapidly converges to the *answer* so that the computer can produce function values with minimal CPU time. Rapid convergence means that the term $|answer - sum\ so\ far|$ rapidly gets smaller. If the terms in the series are shrinking montonically and sufficiently fast (e.g. like a power series) then checking that the current term is small is equivalent to checking that $|answer - sum\ so\ far|$ is small. This is useful because you don't know the value of *answer* up front!

A related problem is expressions for the value of $\pi$. This has a long tradition back to the ancient Greeks and Babylonians ($\pi = 3\frac{1}{8}$ good to about 2 figures). A famous expression is $\pi^2/6 = 1 + 1/4 + 1/9 + 1/16 + ....$ Euler (1735) first showed what the series sums to $\pi^2/6$. However, its convergence is very slow. A much more efficient expression attributed to Euler is:

$$\pi/2 = 1 + \frac{1}{3} + \frac{1}{3}\frac{2}{5} + \frac{1}{3}\frac{2}{5}\frac{3}{7} + ... \tag{1}$$

A key feature of this series is that each term is more than a factor of 2 smaller than the previous term. This means it converges nicely and gives you an extra binary bit more accuracy for every term added. An algorithm for summing this series to a specific accuracy is given below.

**Algorithm: Estimating Pi**

① [ Set-up ] First term t = 1, Initial sum s = t, term counter n = 1

② [ Increment term counter ] n = n + 1

③ [ Calculate next term ] New t is old t multiplied by (n-1)/(2n-1)

④ [ Add term to sum ] Add t to s

⑤ [ Are we close enough? ] If t is small we are done: $\pi$ estimate = 2 s

⑥ [ Loop ] go to ② for next term

**1)** Write a program that calculates $\pi$. The program should read in the smallest term to include in the sum. That is, if the last term added is less than or equal to this value, we consider it small and the current partial sum is used to estimate pi. Use the skeleton provided `pi.cpp` as a starting point. At a minimum you should add code wherever there are elipses '...'. **NB:** If your program never stops when you run it, it may have a bug or some other problem. In this case use **control-C** to stop it.

I have included a Makefile in /home/2G03/HW4 to *make pi* that uses the **-O0** compiler option (no optimization). This is important because the compiler may otherwise choose to use a different precision to the one you specified and the behaviour will be harder to interpret.

Please go to `cd ~/HW4` where your `pi.cpp` is and run: `submitPi` to confirm that your program is working correctly before continuing with the following questions.

**2) Compile your program with `make pi` and run it to answer this and the following questions.** What is the pi estimate where the smallest term you include is the first one below 0.01 (1e-2) ? Write down the last term included and how many terms in total.

**3)** What is the pi estimate where the smallest term you include is the first one below $10^{-4}$ (1e-4) ? Write down the last term included and how many terms in total.

**4)** Do the previous two results support the assertion that the last term included indicates the error in the estimate?

**5)** What is the pi estimate where the smallest term you include is the first one below $10^{-20}$ (1e-20) ? Write down the last term included and how many terms in total.

**6)** Does this last result support the assertion that the last term included indicates the error in the estimate? Why? (This relates to the discussion of precision from class and assumes you declared each of sum, term and small as a standard float type as in the skeleton code)

*file: pi.cpp*

```
#include <iostream>
#include <iomanip>

int main()
{
// Program for estimate Pi to a given accuracy
// Filename pi.cpp Src: JWW 2020

  int n;
  float term, small, sum, pi;

  std::cout << "Welcome to the pi estimator program\n";

  std::cout <<  ... ;
  std::cin >> small;

  n = 1;
  term = 1;
  sum = term;

  for (;;) {
    n++;
    term = ...;
    sum = ...;
    if (term <= small) break;
  }

  pi = ...;

  std::cout << std::setprecision(20) << "Our pi estimate is " << pi << "\n";
  std::cout << "This differs from true pi by " <<
    pi-3.14159265358979323846264338327950288419716939937510582097494 4e0 << "\n";
  std::cout << "The smallest term included was " << ... << "\n";
  std::cout << "The number of terms summed was " << ... << "\n";
}
```

**7)** What does the algorithm do if you enter 0 for small? Why?

**8)** What does the algorithm do if you enter -1 for small? Why?

**9)** Correct your program to check for valid input and hand this version in. Use an if then expression (similar to the code below) to appropriately avoid the problems with certain values of the small variable.