Algorithm to compute π

Question 1:

Construct an algorithm in C++ to approximate π. Assume that you are working on a project with a small development team and your algorithm will be used as a part of a larger code base.

- Assume that the required precision of the calculation is not known until run-time.
- You do not need to determine the error $|x - \pi|$ associated with your calculation.
- Your algorithm should be able to work with single or double-precision floating point numbers
- Assume you have access to a compiler that supports any version of C and/or C++ that you would like numbers.

You can use any method, libraries etc. that you would like with the exception of functions that approximate transcendental numbers (e.g. std::asin, M_PI or similar).

It might be useful to know that $\pi = 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$ (known as the Gregory Series). The first few terms in the series are $4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$.

This is not the only way to compute π and there is no special reason why this method should be preferred over any other method.

You do need to write a Make file, compiler/linker commands or explicitly write out #include statements, but you are responsible for appropriately using namespaces.

Question 2:

Suppose your team notices that your algorithm is a performance bottle-neck, how would you speed it up?

Question 3:

Suppose someone else in your team has written an algorithm to compute π using the Gregory Series $\left(\pi = 4\sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{2k-1}\right)$.

Their algorithm, in pseudo-code, takes the following form:

Algorithm 1 Estimate π using Gregory Sum formula, using pairs of terms.

Input: $n \in \mathbb{N}$ ▷ Specify the number of terms in the series – must be an even number

1: $x \leftarrow 0$ ▷ Initialize the total sum to zero.
2: **for** $k \in [1, 3, 5, \dots, n/2 - 1]$ **do**
3: $\quad x \leftarrow x + \left(\frac{1}{2k-1} - \frac{1}{2(k+1)-1}\right)$
4: **end for**

return $4x$

The first few terms using this algorithm would read $4((1 - \frac{1}{3}) + (\frac{1}{5} - \frac{1}{7}) + \dots)$.

You notice that after a large number of iterations this algorithm exhibits some strange behavior – it does not seem to converge to π. You decide to investigate further and notice the error term $|x - \pi|$ does not go to zero. What might be causing this problem?

Question 4:

Suppose your team decides that the number of iterations/recursions necessary to guarantee required precision can be determined at compile time. Further it has been determined that double-precision floating-point numbers are the only type that needs to be supported.

How would you re-write your algorithm to be computed at compile time instead of run time?