Aaron Rosenthal
CS430
Group 4-E
HW4

Problem Overview:

The goal is to calculate the value of pi over a number of iterations by using the summation:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

This problem is very simple to solve iteratively; the total can be stored in a variable and the current iteration can be used to calculate the next value to be added or subtracted, with a modulus to determine whether to add or subtract. Once the specified iteration is reached (i.e. the value of k equals the input to the function), the total can be multiplied by 4 and the resultant can be returned as the answer. Our implementation skipped this final step by using 4 in the numerator rather than 1 during calculation of the sum.

The only necessary function was 'double calcPi(int numIter)' which would take a number of iterations as input and output a double containing the result of the summation for the value of *k* equal to 'numIter'. This function was implemented in 'pi.c' with the associated header file 'pi.h'.

Our implementation iterated from 0 to numIter-1, but it didn't calculate the next value to add based on the current value of numIter. Instead, it followed a pattern of simply incrementing the denominator of the next value to be added by 2, which was easily derived from the Wikipedia page.

A driver class -- 'piDriver.c' -- was implemented in order to allow for the testing suite and the actual program to both be built by the same makefile. The driver class simply takes an input -- or sets a default if none are present -- and calls calcPi() with it.

Overview of Test Plan:

Testing was also simple; using free online math tools or a calculator, we could determine expected values of pi for any given input *k.* Then we could use googletest's ASSERT_DOUBLE_EQ function to determine if calcPi() had returned the expected value.

Code Review:

The only problem I saw with the code was a couple of erroneous comments. For example, calcPi() has a comment suggesting it can take either an int for number of iterations or a double for tolerance, but it can only take the int for iterations.

The code itself was easy to follow. I didn’t see any glaring flaws.

Reflection:

The only gripe I had with the carousel assignment was that when it came time to implement tests and the actual problems, the workload was incredibly uneven. Making tests for calculating pi was much less involved than testing MM-Mult, and implementing Jacobi2d took much longer than the Fibonacci sequence. I’m unsure how this process could be made any more fair, however.