CS120 - Computer Science I
Lab #4
Fall 2014

Due: At the end of lab.

The purpose of this lab is to use loops to calculate approximations of some standard mathematical values. Irrational numbers, such as π and e, can only be approximated using mathematical formula. In this lab we'll write two different programs to calculate two approximations.

1 π

Begin by creating a new program called `piLab4SectionX.cpp` (where `X` is your section number). As always include your name, date, etc. in a comment block at the beginning.

Values like π can be approximated using an infinite sum. Each term in the sum moves the total a little closer to the goal value. One of the simpler formulas to approximate π is:

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

(Other formula can be found at: http://mathworld.wolfram.com/Pi.html.) Using this formula requires a loop that increase k from 1 to some upper bound. At each step of the loop the next term in the sum is added to a running total. Your output should show both the value of k and the current approximation. For example, the first 4 approximations are:
k π
1 4
2 2.666
3 3.466
4 2.895
To test your program use a large upper bound and check that the total is approaching π.

When you are done with the program submit it using cscheckin.

2 e

Create another new program called `eLab4SectionX.cpp` (where `X` is your section number). As always include your name, date, etc. in a comment block at the beginning.

One of the simpler formulas to approximate e is:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} \tag{2}$$

(Other formula can be found at: http://mathworld.wolfram.com/e.html.) As with π using this formula requires a loop that increase k from *0* to some upper bound (note that the starting bound is now 0). At each step of the loop the next term in the sum is added to a running total. However, the sum includes the *factorial* of k, which is defined as:

$$k! = 1 * 2 * 3 * ... * k \tag{3}$$

with

$$0! = 1 \tag{4}$$

In order to calculate the factorial you will need to use an *nested* loop that counts from 1 to k.

Your output should show both the value of k and the current approximation. For example, the first 4 approximations are:
k e
0 1
1 2
2 2.5
3 2.666
To test your program use a large upper bound and check that the total is approaching e. However, the factorial grow large very quickly, so if your upper bound is too large you will get an overflow error (the value of $k!$ will be too large to be stored).

When you are done with the program submit it using cscheckin.