CMPE50 Object-oriented Concepts and Methodology, Spring 2020, Tarng

**Instructions**: Before you leave the lab, you should submit your answers through Canvas Assignment Lab1. Please submit your answers (*.cpp* files with appropriate documentation/comments/**output**) even if you couldn’t complete/run them. After the submission you need to demonstrate the code to ISA by the next lab in order to get graded.

**Lab #1: C++ basics, data types, simple I/O, control structures**

1. Check out the Eclipse installation on the lab PC to ensure it is working and create a workspace of your own.
2. Create an Eclipse project and write your first C++ program to print the following sentence: “*A fool thinks himself to be wise, but a wise man knows himself to be a fool.*” Copy the output from the console and copy it to the .cpp file in a comment section at the end of the code. Add some description such as “The output of the Lab Exercise 1.2.”

*(Hint: Create a “Hello World C++ Project” as the base and modify it.)*

1. Create an Eclipse project and write a complete program that declares an integer variable, reads value from the keyboard into that variable, and writes to standard output the square of the variable 's value. Copy the output to a comment section in the .cpp file.
2. Create an Eclipse project and write a complete program that reads in an integer that is greater than 2 (let's call it *k*) and finds and prints all of the prime numbers between 3 and *k*.  A prime number is a number such that 1 and itself are the only numbers that evenly divide it (for example, 3, 5, 7, 11, 13, 17, ...).

One way to solve this problem is to use a doubly nested for-loop. The outer for-loop can iterate from 3 to *k*while the inner for-loop checks to see if the counter value for the outer loop is prime. One way to see if number *n*is prime is to iterate from 2 to *n*-1 and if any of these numbers evenly divides *n*, then *n*cannot be prime. If none of the values from 2 to *n*-1 evenly divides *n*, then *n* must be prime. (Note that there are several easy ways to make this algorithm more efficient.)

*Question: How do you check if an integer k evenly divides another integer i? (Hint: Use the modulo operator “%”. See the algorithm and pseudo code below.)*

Input Details: The input consists of just one integer which you may assume is 3 or greater. There is no prompt.

Output Details: Each prime number is printed at the beginning of a new line and is followed by the predicate "is a prime number."

**Do not forget to copy and paste the console output to the source code file in a comment section.**

## Algorithm for determining if an integer k is prime (primality test)

1. If the k is 2 or 3 it is a prime. Stop.
2. Iterate from i = 2 to i = (k-1), test if k can be evenly divided by i, if so k is not a prime, stop.
3. If k passes the tests in step 2, then k is a prime.

## Pseudo-code for primality test

1. int i, k
2. bool is\_prime = true
3. if (2 == k || 3 == k)

{

k is prime. stop

} else continue step 4

1. for (i = 2; i <= k-1; i++)

{

if (0 == k % i) // check if k can be divided by i

{

is\_prime = false

stop the iteration

}

}

1. if (is\_prime), then k is prime