**Lab 15 – Prime Checker**

Open BlueJ, and create a new BlueJ project titled **Lab15-PrimeChecker** in your CS\LABS folder.

Create a new class with this code skeleton:

//Name:

import java.util.\*;

public class PracticeProblems

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

Similar to while loops, for loops run a block of code more than once. The difference, though, is that for loops generally run a block of code *for* a set number of times.

**Before each problem, insert a COMMENT with the problem number.**

1. Write the code, using a for loop and a loop variable, to produce the following output.

123456789

1. Write the code, using a for loop and a loop variable, to produce the following output.

1 2 3 4 5 6 7 8 9

1. Write the code, using a for loop and a loop variable, to produce the following output.

2 4 6 8 10 12 14 16 18 20

1. Write the code, using a for loop and a loop variable, to produce the following output.

20 21 22 23 24 25

1. Write the code, using a for loop and a loop variable, to produce the following output.

20 30 40 50 60 70 80 90

1. (Riddle) How far can a dog run into the woods?
2. Write the code, using a for loop and a loop variable, to produce the following output.

0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10

1. Get a value for an integer called numfrom the keyboard. Write a for loop to print numasterisk characters. The example below assumes the user types an input of 11.

\* \* \* \* \* \* \* \* \* \* \*

1. Write the code, using a for loop and a loop variable, to produce the following output.

1 4 9 16 25 36 49 64 81 100

1. Write the code, using a for loop and a loop variable, to produce the following output.

1 2 ? 4 5 ? 7 8 ? 10 11 ? 13 14 ? 16

1. Get the value of an integer n from the keyboard. Write the code, using a for loop, to print the value of n! (n factorial). Note – factorials get very big very fast, so n should be limited to a single digit.
2. Complete the “Worksheet – For Loops” Excel document. Don’t forget to save when finished.

**Prime number checking app**

Create a new class with this code skeleton:

//Name:

import java.util.\*;

public class PrimeChecker

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

A prime number is a number that is ***only evenly divisible by 1 and itself***. Recall that the modulus operator (%) calculates the remainder of the division between two numbers.

First, prompt the user to enter an integer. Create a program that will check if that number is prime and print to the console (screen) the result. Check for a prime number with a for loop, and put the entire program in a while loop (to allow users to continue checking numbers until they choose to exit).

Sample output (**user input shown in red**):

Enter 1 to check if a number is prime or 2 to exit >>> **1**

Please enter an integer to check >>> **13**

13 is a prime number

**(Advanced) Counting primes**

After you have the basic algorithm for checking if a number is prime, think about how you can optimize your prime number checking loop.

What if you wanted to check **how many primes** there were from 1 to 999,999\*? That would require nested loops, and would take a really long time!

*\*Note – 1 is not considered a prime number*

Are some numbers *ever* going to be prime? Do you need to check all the way from 1 to the number itself? Test it with a large number (use a long data type number). How you check the time your program takes to run is shown below:

double start = System.currentTimeMillis(); //get starting time

/\*

\* code to check if a number is prime goes here

\*/

double end = System.currentTimeMillis(); //get ending time

System.out.println(“Run time = ” + (end – start)/1000 + “ seconds”);