Lab 5 (25 points total)
The purpose of this lab is to apply practical experience to Chapter 5 concepts. There are several learning objectives to this assignment

- Using loops to solve problems (for, while, do while)
- Introductory file input and output
- Breaking problems up into logical function areas by defining methods
- Incorporating **javadocs** for **each method** as well as the use of **@param** and **@return tags as required**

Falling Distance (Prog Chall5) – pg311 (5pts)
Ensure that your output is neatly lined up. For this program, you should have two methods, the main() and getDistance() that has an int parameter for passing in seconds and that returns a double. Since g is a constant, you can define ACCELERATION as `public static final double ACCELERATION = 9.8;` Ensure the output distance value is limited to **two decimal places**. A prompt should ask the user how many seconds to compute. You can use this number in a for loop to run the controlling expression. distance=$0.5*g*t^2$ where g=9.8. Hint: You can use Math.pow(numOfSecs,2) for t^2 **//also, note 0.5 to prevent multiplying by 0 (1/2->0).**

```
Please enter how many seconds to compute: 10

Time(secs)  Distance(m)
=========   ===========
1           4.90
2           19.60
3           44.10
4           78.40
5           122.50
6           176.40
7           240.10
8           313.60
9           396.90
10          490.00
```

Prime Numbers (Prog Chal13) – pg315 (5pts)
Your program should tell the user whether the original number is prime or is not prime. Also ask the user whether they want to calculate another number after each run. Hint: the 2nd part (ask to calculate another number) is a do while loop. In addition to your main(), you should have a 2nd method called testForPrime that returns a boolean value and has a whole number (int) assessed. You can use either console or JOptionPane or a combination for input and output to the user.

Test at least one number that is a known non-prime(ie 8) and one number that is a known prime (ie 19)

```
Enter a whole number>2 (ie 19) to test if prime
8
8 is not prime
Enter yes to test another number, no to quit
yes
Enter a whole number>2 (ie 19) to test if prime
19
19 is prime
Enter yes to test another number, no to quit
no
```

Grade Processing (15pts)
Write a program that reads values from a text input file. For each line of numbers you will determine the number of As, Bs, Cs, Ds, Fs, the lowest score, the highest score, and the average. You will write the output to a file. Each of the method details are explained below.

Your program will include three methods

1) getInFile
    a. returns a String for the path name to the input file
    b. does not have any method parameters
    c. uses JOptionPane to prompt the user to provide a path / file name. NOTE-If you put the input file in the same directory (folder) as Grades.java, you can just enter the file name (input.txt) vs having to include a path.
    d. should test to see if the file can be found. If a file cannot be found based on the path / file name, you should tell the user that the input file cannot be found and prompt the user for a new file path. Hint: see page 245 (lines14-21),yup a while pit 😊
    e. Since you will be working with FileIO, you need to throws IOException (passing the buck) to main() that will also need to throws IOException since we will not be try/catching the exception. This will be later in the course
2) getOutFile
    a. returns a String for the path name to the output file such as output.txt since PrintWriter will be creating a new file in the processFile()
    b. does not have any method parameters
    c. uses JOptionPane to prompt the user to provide a path / file name
3) processFile
    a. receives two String arguments, one for inFile and one for outFile, requiring two String params
    b. Since you will be working with FileIO, you need to throws IOException (passing the buck) to main() that will also need to throws IOException since we will not be try/catching the exception. This will be later in the course
    c. reads in each line of the input file (.hasNextLine() in a while pit) and processes scores (.nextInt() internal while pit) until sentinel value (-1) reached. NOTE: It is possible for a line to just contain -1 meaning no grades to compute for that line. For each line in the input file, writes the results to a file. The file should contain output like below. **Pitfall-Make sure you cast (double) to avoid XX.0 all the time.** Your output file should be set up such that each time the program runs a **NEW file is created and NOT appended to. In other words, you can use PrintWriter with a String var vs. FileWriter var argument.**

```
Set 1 of grades calculated
Number of As: 4
Number of Bs: 4
Number of Cs: 4
Number of Ds: 4
Number of Fs: 2
The high score was: 100
The low score was: 40
The avg score is: 76.1

Set 2 of grades calculated
No grades to average

Set 3 of grades calculated
Number of As: 5
Number of Bs: 4
Number of Cs: 3
Number of Ds: 2
Number of Fs: 1
The high score was: 100
The low score was: 52
The avg score is: 82.1
```

Submitting your work

For all labs you will need to provide a copy of all .java files. **No need to provide .class files. I cannot read these.** ***NOTE – For Replit, please update Main.java to another name such as TempProb.java, ProChall3.java, etc.*** In addition to your .java files, you will need to provide output files of your console. The name of the output file should match the class name and have the .txt extension such as TempProbOut.txt, ProChall3Output.txt. For GUIs such as JOptionPane, you will instead need to create screenshots. For Windows users, Snipping Tool is a great way to do this. Chromebook - Shift+Ctrl+Show Windows. Mac OS users, you can see how to take screenshots using the following url - https://support.apple.com/en-us/HT201361.