Computer Science 111 Intro to Algorithms and Programming: Java

Programming Project #1 – Algorithm Writing and Simple Programming Exercises (30 points)

Due 3/2/20

Part I

Algorithms are precise step-by-step instructions that describe how to accomplish a desired task. A formal algorithm for use with computers or in mathematics must be very detailed and must resolve the ambiguities that we, as humans, take for granted in everyday life. The first exercise in Project 1 involves writing some algorithms.

Example Problem: Write an algorithm to compute and output the area of a circle with a radius of 1.50.

Example Solution:

```
01 Steps for CircleArea with Radius, PI and Area
02 Set the value of PI to 3.14159
03 Set the value of Radius to 1.50
04 Set the value of Area to PI * Radius * Radius
05 Output the value of Area
06 Stop
```

The above is an example of a pseudocoded algorithm which describes the process of computing and displaying the area of a circle with a radius of 1.50. Notice that every step represents one instruction and that all quantities are represented by names such as **Radius** and **Area**. All quantities in the algorithm are "given" specific values and nothing is left up to assumption. Additionally, all of the steps are written in a logical sequence and in such a way that a human being should be able to execute this list of instructions and correctly compute the **Area** of a circle with radius 1.50.

TriangleArea (4 points): Using any text editor, write a pseudocoded algorithm named **TriangleArea**, which calculates and outputs the **Area** of a triangle with a **Base** value of 3.5 and a **Height** value of 4.85. The formula for the area of a triangle is: $0.5 \times Base \times Height$. Save your algorithm as a **plain text file** named **Algorithm1.txt**.

MileToKilo (5 points): After you have written an algorithm, you must test the algorithm to verify that the step-by-step instructions achieve a *correct* result or *correctly solves the problem*. Testing an algorithm involves following the steps of your algorithm *exactly* as they are written, and performing the algorithm's computations on paper. If, in the end, your algorithm consistently yields a correct result when testing it, then you have some indication that you have devised an effective solution for the stated problem. The second exercise in Part I involves testing an algorithm. Testing an algorithm is also known as "desk checking" that algorithm.

The following algorithm should compute and display the number of kilometers in 100 miles. But as they are written, the steps of this algorithm contain two errors:

```
01 Steps for MileToKilo with miles, kilometers and KILOMETERS_PER_MILE
02 Set the value of KILOMETERS_PER_MILE to 1.609
03 Set the value of kilometers to the result of miles * KILOMETERS_PER_MILE
04 Stop
```

Test the algorithm, determine the two errors, and then rewrite the algorithm so that it works as intended. Copy the algorithm as it is written here, insert the two missing steps then renumber the steps. Save your rewritten algorithm as a **plain text file** named **Algorithm2.txt.**

Part II

This portion of the project assumes that you have watched me demonstrate the process of starting a Netbeans project in lab. If you have not watched this demonstration, STOP, and ask me for help. Additionally, make sure you read section 1.11 in your textbook titled “Developing Java Programs Using Netbeans” prior to starting work on this project.

Create a folder on your flash drive or your laptop’s hard drive for ALL of your Java projects. Something like: **e:\JavaProjects** or **c:\JavaProjects**, although your drive letter will probably be different. If you are using a lab computer hard drive, you MUST save the code to external storage when you are done.

DisplayAPattern (4 points): Create a Netbeans project named **Project1**. Within **Project1**, create a main class with the name **DisplayAPattern**. In the main method, write println statements which display the following pattern:

```
    J      A     V     V     A
    J    A   A    V   V    A   A
J   J   AAAAAAA    V V    AAAAAAA
 JJJ   A       A    V    A       A
```

Use a sequence of exactly four (4) System.out.println statements to complete this exercise.

PrintATable (5 points): DO NOT create another project, we are going to add a new file to **Project1**. From the Netbeans **File** menu, select **New File**. In the dialog box under **File Type**, select **Java Main Class** (note that you may have to choose **Other**, then find **Java Main Class** in a list). Give the main class the name **PrintATable**, verify that it is part of **Project1** then click the **Finish** button. In the body of the **PrintATable** main method, write a sequence of **System.out.println** statements which display the following table:

```
a    a^2   a^3
1    1     1
2    4     8
3    9     27
4    16    64
```

From the Project Navigator in Netbeans, you can run this new program by right-clicking on the source code file name and selecting **Run File** from the list or if the program is being edited in the Netbeans code editor window, press **Shift-F6**.

MyInitials (8 points): DO NOT create another project, we are simply going to add another new file to **Project1**. From the Netbeans **File** menu, select **New File**. In the dialog box under **File Type** select **Java Main Class** and click **Next**. Create a main class with the name **MyInitials**, verify that it is part of **Project1** and click the **Finish** button. In the main method of the **MyInitials** class, write a sequence of **System.out.println** statements to display your initials (2 – 3 letters) using an 8 x 8 pattern. For example, here is an 8 x 8 pattern for my initials BR:

```
BBBBBBB   RRRRRRR
B      B  R      R
B      B  R      R
BBBBBBB   RRRRRRR
B      B  R   R
B      B  R    R
B      B  R     R
BBBBBBB   R      R
```

__**Use exactly eight (8) print line statements to complete this exercise**__. From the Project Navigator in Netbeans, you can run this new program by right-clicking on the source code file name and selecting **Run File** from the list or if the program is being edited in the Netbeans code editor window, press **Shift-F6**.

CircleAreaPerimeter (4 points): DO NOT create another project, we are simply going to add a new file to **Project1**. From the Netbeans **File** menu, select **New File**. In the dialog box under **File Type**, select **Java Main Class** and click the **Next** button. Create a main class with the name **CircleAreaPerimeter**, verify it is part of **Project1** and click Finish. In the body of the **CircleAreaPerimeter** main method, write statements which compute and display the area and perimeter of a circle which has a radius of 5.5 using the following formulas. **Assume a value of 3.14159 for π:**

$$perimeter = 2 \times radius \times \pi$$

$$area = radius \times radius \times \pi$$

From the Project Navigator in Netbeans, you can run this new program by right-clicking on the source code file name and selecting **Run File** from the list or if the program is being edited in the Netbeans code editor window, press **Shift-F6**.

When you have completed all of the algorithm writing and programming exercises, let me know. I will sit with you and read your two algorithms. You will then test run your Java programs for me. I will either suggest changes to be made or will give your project a score. After I have given your project a score, I will instruct you on how to zip the project folder up and submit it to Canvas. **<u>DO NOT zip up your project or upload the zip file unless I have instructed you to do so</u>.**