COSC 1046 – Fall 2018 - Assignment 3

This assignment is designed to give you some practice and experience writing Java applications using loops. Each question specifies the name that each file/application should have. Please follow this naming convention. When you have completed the assignment compress all of your files into a single archive using Windows (Right Click folder -> Send To -> Compressed Folder) or OS X (Right Click folder -> Compress). Using a 3rd party compression utility such as WinRAR or 7zip may render your files unreadable and un-markable. Submit a single compressed file to D2L. You can resubmit your files as many times as you would like up to the due date and time.

Be sure to include your name and a brief description of your program (as comments) at the top of each file. Pay attention to using good variable names. If you have any questions, please check or post to the forum.

Submit solutions for these questions. Wherever applicable, do your best to reproduce my output exactly. In my sample **bold** indicates user-input.

If you work in partners, submit only one solution and make sure that both partners' names are in ALL files. If only one name appears, only one person will get the grade. No exceptions.

A word in general on assignment grading (for future assignments): If your program produces the displayed output and meets the criteria specified in the question you should expect to receive full marks. Deductions are taken when there are deviations in the output – small deductions for small deviations (calculation error, improper formatting, etc.) while larger deductions are taken for larger deviations (missing output, substantially incorrect values, etc.). Express your creativity in your code, not in your output. Also, deductions will be taken for:

- lack of comments
- poor variable names
- programs that don't run at all (large deduction), so make sure your program runs, even if it is not complete.

If you have any questions or concerns about the grading scheme before you submit an assignment, please post to the forum and ask for clarification. If you have concerns about the grading you've received on an assignment, please email me and I will review the grading form.

Do all of the questions to ensure that you are practicing all concepts but submit solutions to questions 1 and 2 ONLY for grading. It is assumed that you are doing all of the questions and some of them may be referenced in future assignments. Solutions will be provided only for required questions.

Question 1. (`Stats.java`) Write a program that prompts the user to enter an arbitrary (this means unknown) number of positive double numbers. When the user enters a negative number the input is complete. At this point the program should display the count, maximum, minimum, sum, product, average and range (distance between maximum and minimum) of the numbers. Assume the user enters properly-formatted numbers as input. Hint: most of this work can be done inside of a single while loop.

```
C:\cosc1046\a2\>java Stats
Enter a double number (negative to quit): 1.0
2.0
3.0
4.0
5.0
-1

Count: 5
Max: 5.0
Min: 1.0
Sum: 15.0
Product: 120.0
Average: 3.0
Range: 4.0
Program complete.
```

```
C:\cosc1046\a2\>java Stats
Enter a double number (negative to quit): 10.39
14.219
81.2
0.001
9.983
6.9
12
10012.34
-2

Count: 8
Max: 10012.34
Min: 0.0010
Sum: 10147.033
Product: 9.928133947689667E7
Average: 1268.379125
Range: 10012.339
Program complete.
```

Question 2. (`Nested.java`) Write ***nested for loops*** that will display a multiplication table. Ask for an integer from the user. Continue to re-prompt the user as long as the input is less than 1 or greater than 10. When you get valid input display a table similar to what is shown below.

```
C:\cosc1046\a2\>java Nested
Enter an integer between 1 and 10:-1
Enter an integer between 1 and 10:0
Enter an integer between 1 and 10:4

1     2     3     4
2     4     6     8
3     6     9     12
4     8     12    16
```

```
C:\cosc1046\a2\>java Nested
Enter an integer between 1 and 10:7

1     2     3     4     5     6     7
2     4     6     8     10    12    14
3     6     9     12    15    18    21
4     8     12    16    20    24    28
5     10    15    20    25    30    35
6     12    18    24    30    36    42
7     14    21    28    35    42    49
```

If you look at the output above you'll see that it is symmetric on the diagonal. Try removing the duplicate output so that it looks more like this:

```
C:\cosc1046\a2\>java Nested
Enter an integer between 1 and 10:4

1     2     3     4
      4     6     8
            9     12
                  16
```

Question 3. (Primes.java) Write a program that prompts for an integer input (assume the user enters a positive integer, you don't need to do input checking this time). Compute and display all of the prime numbers up to and including that integer. Remember, a prime number is a number that is only divisible by 1 and itself.

There are lots of ways to solve this problem. Here is pseudo code for one possible solution that describes the process and assumes the user requested all of the primes between 1 and n:

```
boolean primeFlag set to true
For each number i between 2 and n:
        For each number j between 2 and i:
                if the remainder of j / i is not zero
                        set primeFlag to false
        Check the value of primeFlag and print the number if value was prime (ignore otherwise)
```

```
C:\cosc1046\a2\>java Primes
Enter an integer value: 20
Prime numbers:
2
3
5
7
11
13
17
19
```

```
C:\cosc1046\a2\>java Primes
Enter an integer value: 100
Prime numbers:
2
3
5
7
… more numbers not displayed here
983
991
997
```

Two things for these marks. 1. Take two integers as inputs and check only for the prime numbers between the two digits. 2. The pseudo code above works but is VERY inefficient. Make your program more efficient – Don't use primeFlag at all and you don't need to check **all** of the possible divisors so have your inner loop end immediately when you've checked enough of them to ensure the number is NOT prime.

```
C:\cosc1046\a2\>java Primes
Enter two integer values: 10 20
Prime numbers:
11
13
17
19
```

Question 5. (Guessing.java) Write a Java application that simulates a guessing game.

Variation 1: Have the application select a random number between 1 and 100. Then prompt the user to enter a "guess" integer. If the user guesses correctly, print a message and end the program. If the user is not correct print "too high" or "too low" appropriately and allow the user to guess again. If the user does not guess correctly within 10 guesses, have the program end.

Variation 2: Same as variation 1 but allow the user to continue guessing until they get it right. Display the number of guesses it took before the program ends.

Variation 3: Similar to variation 2 BUT have the user enter the number-to-be-guessed and have the computer guess the number. There are efficient ways to do this and not-so-efficient ways. Can you figure them out?

Question 6. (TwoDice.java) Write a program that prompts for an integer input. While the input is less than 1 re-prompt and re-collect from the user. The program should simulate two separate dice (use random numbers with the appropriate range) being rolled as many times as was requested by the user. When the program is done, output the percentage of rolls that resulted in a sum of 7, a sum of 2 and a sum of 11. Your output will not necessarily match mine in this case.

```
C:\cosc1046\a2\>java question6
Please enter a positive integer: -1
Please enter a positive integer: -9
Please enter a positive integer: 10
Sevens: 30.0%
Twos: 10.0%
Elevens: 10.0%

C:\cosc1046\a2\>java question6
Please enter a positive integer: 1000
Sevens: 14.899999999999999%
Twos: 2.6%
Elevens: 5.1%

C:\cosc1046\a2\>java question6
Please enter a positive integer: 1000000
Sevens: 16.6575%
Twos: 2.7775000000000003%
Elevens: 5.5666%

C:\cosc1046\a2\>java question6
Please enter a positive integer: 100000000
Sevens: 16.663733999999998%
Twos: 2.7761899999999997%
Elevens: 5.5555449999999995%
```

For the curious…See how well your program compares to 2x dice doll - Mathematical Probabilities

2 - 2.78%
3 - 5.56%
4 - 8.33%
5 - 11.11%
6 - 13.89%
7 - 16.67%
8 - 13.89%
9 - 11.11%
10 - 8.33%
11 - 5.56%
12 - 2.78%

Keep the same user input but in addition to doing what was requested above, simulate 5 more dice. Roll all five dice and keep track of the percentage of time that all five dice are the same (as in a Yahtzee!).

```
C:\cosc1046\a2\>java question6
Please enter a positive integer: 100000000
Sevens: 16.670704%
Twos: 2.780491%
Elevens: 5.549894%

Yahtzees: 0.07684099999999999%
```

Again, for the curious, the odds of rolling a yahtzee in a single roll is 1/1296.

Question 7. (IntChecker.java) Write a java script that determines whether a user-entered **integer** is odd or even and whether it is evenly divisible by 3, 4, both 3 and 4 or neither 3 nor 4 (see my output below for clarification). Assume the user will enter a positive integer. The output from my solution is shown below. Do your best to duplicate my output. Below is my output from running the program 5 times.

Use boolean operators (&& or ||) in your solution.

```
C:\Users\aaron\Desktop>java question7
Enter an integer:12
12 is even.
12 is divisible by 3 and 4.

C:\Users\aaron\Desktop>java question7
Enter an integer:9
9 is odd.
9 is divisible by 3.

C:\Users\aaron\Desktop>java question7
Enter an integer:16
16 is even.
16 is divisible by 4.

C:\Users\aaron\Desktop>java question7
Enter an integer:13
13 is odd.
13 is neither divisible by 3 nor 4.

C:\Users\aaron\Desktop>java question4
Enter an integer:132
132 is even.
132 is divisible by 3 and 4.
```

Question 8. (`StringOrder.java`)

Write a program that collects three Strings from the user. Display the three strings in alphabetical order regardless of the order in which they were input.

Use boolean operators && and/or || in your solution.

(Hint: If you are having a hard time getting this to work with Strings try it with integers first. The logic is the same but the syntax is a bit different.)