CSE 174 - Fall 2019

PROGRAM #4: 30 points – Due Sunday, September 22, by 11:59 p.m.

Outcomes:

- Write programs that obtain user input
- Write programs that work with and manipulate Strings
- Write programs that compute mathematical results
- Write Java programs that use conditional expressions
- Format and comment source code that adheres to a given set of formatting guidelines
- Use a zip compression tool to combine multiple files

Scoring:

At a bare minimum, the program you submit must have the assigned source code, and your source code must compile and run without crashing.

- If you do not submit your source code (.java files), your score will be zero.
- If you submit source code that does not compile, your score will be zero.
- If you submit source code that roughly resembles the requirements and it compiles, but it crashes under normal operating conditions (nice input from the user), your score will be reduced by 75%.
- Deductions will be made for not meeting the usual requirements (properly formatted source code, class names that do not meet specifications, and so on).

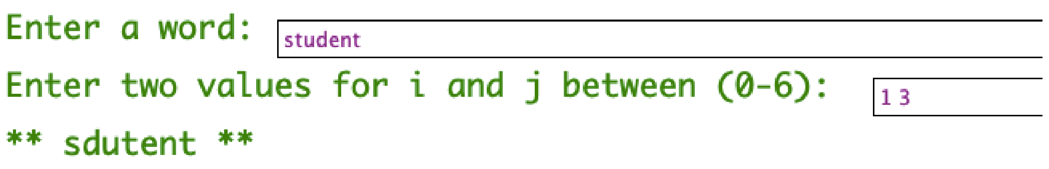
	Full credit	**No credit or Partial credit**
Write solutions to the four given problems (20 points)	The programs you submitted solved the specified problems.	The programs you submitted do not solve or only partially solve the specified problems.
List test cases for the `Donut problem` (5 points)	You thoroughly tested your `DonutBoxes` class, listing the test cases that do and do not work.	You did not list test cases, or missed important test cases.
Format output as specified (5 points)	Your output is formatted as specified, including proper spacing, spelling, rounding values to the specified number of places, and so on.	You did not follow some or all of the requirements for output.

Requirements: You will write the following Java programs

1. **(5 points)** Read and understand the pseudocode of problem R2.19 in java from the end of chapter 2 in your textbook. In a Java class named **`WeekDays`**, write a java program that reads an input int between (0-6), and displays the corresponding day for that given input as below:

```
Enter a number (0-6): 2
** 2 is Tue **
```

2. **(5 points)** Read and understand the pseudocode of problem R2.20 from the end of chapter 2 in your textbook. In a Java class named **`SwapLetters`**, write a java program that reads an input string, 2 indices of that string and displays a new string with the characters at these 2 positions swapped.



```
Enter a word: student
Enter two values for i and j between (0-6): 1 3
** sdutent **
```

3. **(5 points)** Write a program called **`IntGenerator`** that generates a random int value between [50 – 100] (both numbers are included) and displays whether or not the number is ODD or EVEN.

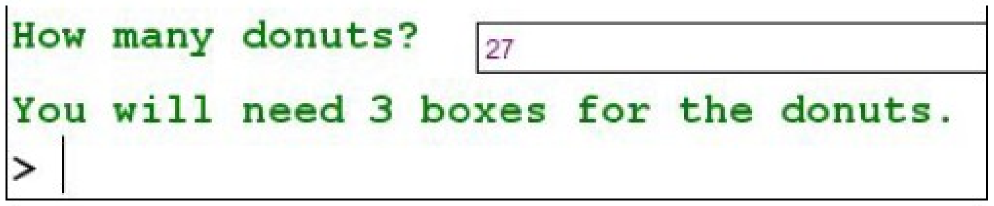
   Sample output 1:

   The generated number is 62 and it is EVEN!

   Sample output 2:

   The generated number is 71 and it is ODD!

4. **(5 points)** A donut shop only has boxes that hold 12 donuts. So, if you order 16 donuts, they will give you 2 boxes. One will be full, and one will have the remaining four donuts. In a class named **DonutBoxes**, write a program that asks how many donuts a customer purchased, and then print the number of boxes needed, formatting your output to match the following as closely as possible.



```
How many donuts? 27
You will need 3 boxes for the donuts.
> |
```

There are a variety of ways to solve this problem. Your goal is to find a solution that only involves simple arithmetic (the symbols + - */ %) . In order to earn full credit, you should have a solution that does NOT use:

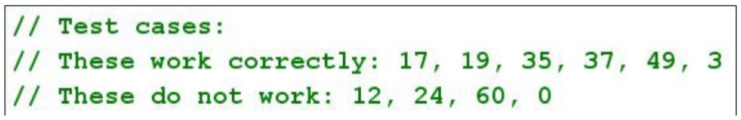
- if statements
- loops
- Methods in the Math class (such as Math.sqrt())

If you write a correct solution but it uses one or more of the ideas from this list, you will
earn partial credit for your solution.
If your program mostly gives correct answers, and you note that in your comments

when you do your testing, you can earn 4 out of the 5 points for this problem.

5. (5 points) Test your **DonutBoxes** class thoroughly. You may find that your program works well with some numbers, but not others. (You only need to test your program with positive whole numbers.) If it does not work correctly with certain numbers, then you should try a different approach. In the **DonutBoxes** class, at the top of the comments, include a list of the numbers of donuts you tested with your program, indicating which test cases worked, and which cases did not. It is important to test thoroughly. If you find that your program does not work with certain numbers, you will earn credit for including the numbers that don't work. If you say that your program works with all numbers, but it does not, you will lose credit for this part of the assignment. You should indicate at least 10 numbers that you tried. Be selective. The goal is to test and find mistakes. Here is a sample of what your source code might say (these comments should appear before the rest of your source code):

Note that this part of the assignment is worth 5 points. You can get full credit for this,
even if your DonutBoxes class does not always produce the correct answer.



```
// Test cases:
// These work correctly: 17, 19, 35, 37, 49, 3
// These do not work: 12, 24, 60, 0
```

6. Follow these steps to submit your work:
   a. Create an empty folder named program4
   b. Put a copy the four source code files (.java) in the program4 folder.Do not put any other files in that folder. There should be exactly 4 .java files
   c. Compress the folder to create a program4.zip file. Be sure it ends with .zip.
   d. Submit only the zip file to the Canvas website.

Note: If you submit your work and decide to modify one of your programs, you need to resubmit a new zip file containing the source code file. Do not rename your source code file. Do not rename the zip file. Canvas may add a number to the name of your zip file. That is fine. But you should keep all filenames the same.