PL05

Due Feb 21, 2019 by 3pm **Points** 10 **Submitting** a file upload **File Types** zip
Available Feb 19, 2019 at 3pm - Feb 21, 2019 at 3pm 2 days

This assignment was locked Feb 21, 2019 at 3pm.

Study Chapter 7 "Recursion".

Enter at least one interview question into the bank of "Interview Questions" - use your alias instead of your name. Find a question on the internet that pertains to any topic that we studied in this class.

IN PREPARATION FOR THE LAB:

1. Load Lab05 files to IDEA, look for **TODO PRELAB** comments in `CountUpDown.java`, and implement the requested methods based on Figures 7-3 and 7-4 (from the textbook):
   - trace the recursive calls for `countDown(5);`
   - trace the recursive calls for `countUp(5);`
   - do not forget to submit `CountUpDown.java` with the PRELAB
2. Analyze the code and the test drivers provided for the remaining classes, and make yourself familiar with the project descriptions.
3. Run the **Maze Solver applet** as shown in the lab description
4. Show steps for converting `345` in base `10`
   - to base `8`
   - to base `2`
5. If you have two numbers 0 and 1, and three positions to place them, the following eight combinations are possible: 000, 001, 010, 011, 100, 101, 110, 111. Design a recursive algorithm that would generate and print each permutation.
6. In the lab you will write **four different recursive** methods that each compute the sum of the integers in an array of integers where the elements in **all solutions** are **added in the order of their position in the array**. Each method however uses a different technique of processing the elements:
   - `sumArrayStartWithFirst` – adds all the elements considering the first element and recursively considering the rest
   - `sumArrayStartWithLast` – adds all the elements considering the last element and recursively consider the rest
   - `sumArraySplitInTwo` – *"divide and conquer"* solution – divides the array in two pieces and processes each of the pieces separately. In this solution computed middle element must be included in the left "half" of the array
   - `sumArraySplitInTwoAddMid` – *"divide and conquer"* solution – divides the array in two pieces and processes each of the pieces separately. In this solution however, the middle element must be excluded

from both “halves”, instead you recursively add elements in the left half, add the middle element, and add recursively elements from the right half.

Design algorithms for each method - model your algorithms after the `displayArray` methods given in Segments 7.15 through 7.18:

Trace your algorithms with the following array: `3 7 5 8 2`