**CMP 352 - Programming Assignment #1**

**Fall 2017**

The first programming assignment of the semester is Exercise 20.17 in your textbook. For this assignment, you are to analyze, design, implement, and test your solution. You and your teammate must provide full and detailed descriptions for each of the items below. Also, your code must compile and run correctly.

**Analysis**

What is the purpose of this program? The purpose of this program is to determine whether a given set of four integers, between 1-13, can be mathematically manipulated to equal 24.

What are the inputs and the expected outputs of the program? This program facilitates a user's input of four integers, between 1-13. Using that input, the program then returns either the mathematical computations that equate to 24, or a simple message that there is no such solution.

**Design**

What data structure(s) will you use to write this program? Describe how your data structure(s) will be used in your program. For this program, we will use multiple Stacks to accommodate the storage and orderly access of the numbers (Integers) which are input from the user, as well as the operators (Characters: \*/+-) that will be used for computation.

Write a short algorithm that describes the steps that your program will follow when executed.

1. The program will prompt user for 4 integers, and save these in an array
2. The program will then generate three arrays
   1. An ArrayList<ArrayList<Integer>> that contains all permutations of all possible values of user input integers
   2. An ArrayList<ArrayList<Character>> that contains all possible combinations of operators (+-\*/)
   3. An ArrayList<String[]> That contains string arrays, each array representing the presence or absence of a parenthesis
3. The program will then combine the above arrays combinatorically into expressions that represent every possibly combination of numbers, operators, and parentheses and save these expressions as strings in an ArrayList<String>
   1. This is accomplished by using nested foreach loops
4. The program will then loop over each expression to test whether it evaluates to 24
   1. This is accomplished via an open-source class called Expression.java, from GitHub
   2. If the expression returns 24, it is added to a list. Before being added it is checked to see if it is already in the list
5. The output of this list is returned to the user, indicating the number of solutions or no solution, and the time taked to reach this conclusion.

**Implementation**

Write the complete Java program for the design that you described above. Do not copy and paste your code on this worksheet. I will review your code in your source file(s).

**Testing**

List at least three different test cases below. List your inputs, your expected outputs, and your observed outputs. Provide screen shots of the console for your inputs and outputs.

Briefly explain why you chose each of the test cases.

**Test Case 1:**

**Input: 2 3 3 4**

**Expected Output: No solutions found.**

**Chosen because the program could not find any output for this.**

**Console Output:**

Enter four integers

2 3 3 4

All 55296 possible arrangements of numbers, operators, and parentheses generated in 0.078 seconds.

Beginning evaluation of these expressions.

Results:

No solutions found. It took 0.727 seconds of my precious time to figure this out for you.

**Test Case 2:**

**Input: 1 4 7 9**

**Expected output: 4 Solutions**

**Chosen because of the difficulty of finding a solution**

**Console Output:**

Enter four integers

1 4 7 9

All 55296 possible arrangements of numbers, operators, and parentheses generated in 0.068 seconds.

Beginning evaluation of these expressions.

Results:

0: (1-9)\*(4-7)

1: (4-7)\*(1-9)

2: (7-4)\*(9-1)

3: (9-1)\*(7-4)

All 4 solutions to puzzle found in 0.697 seconds.

**Test Case 2:**

**Input: 13 13 13 2**

**Expected output: 3 Solutions**

**Chosen because of the difficulty of finding a solution.**

**Console Output:**

Enter four integers

13 13 13 2

All 55296 possible arrangements of numbers, operators, and parentheses generated in 0.081 seconds.

Beginning evaluation of these expressions.

Results:

0: (13-13/13)\*2

1: 2\*(13-13/13)

2: 2\*(13-(13/13))

All 3 solutions to puzzle found in 0.798 seconds.

**Deliverables**

Upload this worksheet and all your source code files to the appropriate Drop Box on Moodle before class time on **Friday, September 22**.