**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. Create 3 Stacks for Characters, and 1 Stack for Integers.

2. Call enterNumbers method to facilitate user input of 4 integers between (1-13). Input is stored in the numbers Stack. If the user inputs integers that are out of range, the method tells the user which integers were out of range and then gives the user a chance to input new numbers. If the user fails again, the method tells the user to start over and stops the program.

3. Call stackNumbers method to populate the numbers Stack with every possible combination of the 4 integers that were input by the user.

4. While() loop begins, and continues until the numbers Stack is empty.

5. Call loadOperatorStacks method to populate the 3 remaining Stacks with the operator Characters (\*/+-). The combination of these Stacks produces a complete list of every possible combination of the operators (\*/+-). The op1 and op2 Stacks each use a nested for() loop. The op3 Stack uses a single for() loop. The loadOperatorStacks method returns 0 to indicate when it's finished.

6. Call combineStacks method and add the returned integer value to the variable "result" each time the method is finished.

7. The combineStacks method begins by pop() 'ing the first 4 integers from the numbers Stack, assigning each of them to a variable (num1, num2, num3, and num4). It then initializes a while() loop which pop() 's the first Character from the op1, op2, and op3 Stacks, assigning them to the variables o1, o2, and o3 respectively. An integer variable carry is assigned, as well as 2 Character variables opar ('(') and cpar (')'). At this point, 3 if() statements cycle through the possible operator Characters (\*/+-) from variables o1, o2, and o3. They are combined with the num1, num2, num3, and num4 variables to perform numerical operations. The result of the first numerical operation is stored as the carry variable. Which is then used for the second numerical operation, whose result replaces the value of the carry variable. This is done a third time, and then another if() statement is used to determine whether the final value in carry is equal to 24. If it is, the entire sequence is submitted to the console on a println that includes parenthesis around the first numerical operation. The variable result is then incremented by one, and the while() loop begins again. This continues until the op3 Stack is empty. At that point, the value of result is retuned to the method call. This completes the first full sequence of numerical operations on the numbers input by the user.

8. The first while() loop continues by calling the loadOperatorStacks method, and then the combineStacks method until the numbers Stack is empty.

9. At this point, the program has either output all the equations that equal 24, or, if the value of result is still 0, the program outputs "No solutions were found." to the console.

10. The program is now finished and stops.

**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:

Inputs: 1 2 3 4

Expected Outputs: Either a list of equations that equal 24, or a message that there are no solutions.

Observed Outputs:

(4\*3)\*2/1 = 24

(4\*3)\*2\*1 = 24

(3\*4)\*2/1 = 24

(3\*4)\*2\*1 = 24

(4\*2)\*3/1 = 24

(4\*2)\*3\*1 = 24

(2\*4)\*3/1 = 24

(2\*4)\*3\*1 = 24

(3\*2)\*4/1 = 24

(3\*2)\*4\*1 = 24

(2\*3)\*4/1 = 24

(2\*3)\*4\*1 = 24

(4\*3)/1\*2 = 24

(4\*3)\*1\*2 = 24

(3\*4)/1\*2 = 24

(3\*4)\*1\*2 = 24

(4/1)\*3\*2 = 24

(4\*1)\*3\*2 = 24

(1\*4)\*3\*2 = 24

(3/1)\*4\*2 = 24

(3\*1)\*4\*2 = 24

(1\*3)\*4\*2 = 24

(4\*2)/1\*3 = 24

(4\*2)\*1\*3 = 24

(2\*4)/1\*3 = 24

(2\*4)\*1\*3 = 24

(4/1)\*2\*3 = 24

(4\*1)\*2\*3 = 24

(1\*4)\*2\*3 = 24

(2/1)\*4\*3 = 24

(2\*1)\*4\*3 = 24

(1\*2)\*4\*3 = 24

(3+2)+1\*4 = 24

(3\*2)/1\*4 = 24

(3\*2)\*1\*4 = 24

(2+3)+1\*4 = 24

(2\*3)/1\*4 = 24

(2\*3)\*1\*4 = 24

(3+1)+2\*4 = 24

(3/1)\*2\*4 = 24

(3\*1)\*2\*4 = 24

(1+3)+2\*4 = 24

(1\*3)\*2\*4 = 24

(2+1)+3\*4 = 24

(2/1)\*3\*4 = 24

(2\*1)\*3\*4 = 24

(1+2)+3\*4 = 24

(1\*2)\*3\*4 = 24

The computations were completed in 3429 milliseconds.

Chosen because it is a basic run of 4 integers.

Test Case 2:

Inputs: 13 12 11 10

Expected Outputs: Either a list of equations that equal 24, or a message that there are no solutions.

Observed Outputs:

(10-11)+12+13 = 24

(10+12)-11+13 = 24

(10/12)+11+13 = 24

(12+10)-11+13 = 24

(12/10)\*11+13 = 24

(12-11)+10+13 = 24

(12/11)+10+13 = 24

(10-11)+13+12 = 24

(10+13)-11+12 = 24

(10+13)/11\*12 = 24

(13+10)-11+12 = 24

(13+10)/11\*12 = 24

(13/10)+11+12 = 24

(11+13)/10\*12 = 24

(13-11)+10+12 = 24

(13+11)/10\*12 = 24

(10+12)+13-11 = 24

(10/12)+13+11 = 24

(12+10)+13-11 = 24

(12/10)\*13+11 = 24

(10+13)+12-11 = 24

(13+10)+12-11 = 24

(13/10)+12+11 = 24

(12+13)+10-11 = 24

(13+12)+10-11 = 24

(12-11)+13+10 = 24

(12/11)+13+10 = 24

(13-11)+12+10 = 24

(12+13)-11+10 = 24

(12\*13)/11+10 = 24

(13+12)-11+10 = 24

(13\*12)/11+10 = 24

The computations were completed in 4473 milliseconds.

Chosen because it is the highest run of 4 integers in range, and the input is from highest to lowest.

Test Case 3:

Inputs: 3 12 5 9

Expected Outputs: Either a list of equations that equal 24, or a message that there are no solutions.

Observed Outputs:

(5-9)+12\*3 = 24

(9\*12)/5+3 = 24

(12-9)+5\*3 = 24

(12\*9)/5+3 = 24

(5+12)-9\*3 = 24

(12+5)-9\*3 = 24

(9-5)\*3+12 = 24

(9+3)/5\*12 = 24

(3+9)/5\*12 = 24

The computations were completed in 4763 milliseconds.

Chosen because it is a scrambled list of 4 integers.

Test Case 4:

Inputs: 6 7 8 9

Expected Outputs: Either a list of equations that equal 24, or a message that there are no solutions.

Observed Outputs:

No solutions were found.

The computations were completed in 3664 milliseconds.

Chosen because it is known not to have any solutions.

**Deliverables**

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