CS3460: Data Structures Fall 2022
Assignment 02: Fun with Stacks
Due Date: Friday, September 16; 23:00 hrs
Total Points: 20

In this assignment, we will first build an integer stack data structure, and then use it to solve a fun problem of computing the molecular mass of a chemical molecule.

1. **Integer Stack (10 points):** An integer stack data structure is a stack that holds integer data and supports two operations.

1. `push(x):` This operation pushes an integer x onto the stack. You are guaranteed to not have more than 100 elements on the stack.
2. `pop():` This operation pops the last pushed element from the stack. We will store only positive integers on the stack, so please return -1 if the user attempts to pop from an empty stack.

You are given the file `IntStack.java`. Please fill out all operations of the integer stack as described above. You may implement the stack as either an array or a linked list. Please test your functions thoroughly by writing an appropriate driver program.

2. **Molecular Mass (10 points):** A molecule can be defined as a sequence of atoms, and represented by a chemical formula consisting of letters denoting these atoms. E.g. letter H denotes an atom of hydrogen, C denotes an atom of carbon, O denotes an atom of oxygen, and the formula COOH represents a molecule consisting of one atom of carbon, two atoms of oxygen and one atom of hydrogen.

To write some formulas efficiently, we use the following rules. Letters denoting some atoms can be grouped by enclosing in parentheses, e.g. formula CH(OH) contains group OH. Groups can be nested, i.e., a group can also contain other groups. To simplify a formula, consecutive occurrences of the same letter can be replaced with that letter followed by a number of these occurrences. E.g. formula COOHHH can be written as CO2H3 and it represents a molecule consisting of one atom of carbon, two atoms of oxygen and three atoms of hydrogen. Furthermore, consecutive occurrences of the same group can be replaced with that group followed by a number of these occurrences. E.g. formula CH (CO2H) (CO2H) (CO2H) can be written as CH(CO2H)3 and molecule represented by both those formulas consists of four atoms of carbon, four atoms of hydrogen and six atoms of oxygen. A number written after a letter or a group is always greater than or equal to 2 and less than or equal to 9. A mass of a molecule is a sum of masses of all its atoms. One atom of hydrogen has mass 1, one atom of carbon has mass 12 and one atom of oxygen has mass 16.

Please write a Java program `MolecularMass.java` that takes a string representation of a molecule as input and prints its total molecular mass. The input is guaranteed to come with only characters C, H or O, and strictly follows the rules given above (that is, one doesn't have to check for the invalidity of the input). It is also guaranteed that there are no more than 100 characters in the input. As a hint, think about how to use the integer stack from problem # 1, and what stack operations are needed. Several sample input and outputs are shown below.

```
Sample Input and Output:

Enter the molecule: H2O
The Molecular Mass of H2O is 18

Enter the molecule: CH(CO2H)3
The Molecular Mass of CH(CO2H)3 is 148

Enter the molecule: ((CH)2(OH2H)(C(H))O)3
The Molecular Mass of ((CH)2(OH2H)(C(H))O)3 is 222
```

Grading and I/O: To simplify the grading process, we will be using automated scripts. You are provided with sample input and output files for most of the programs in class. For this problem, you are provided with `molecularmass.in` and `molecularmass.out`. In order to test your `Java` program, we will use the Unix `diff` utility and execute the following commands.

```
$ java MolecularMass < molecularmass.in > myprogram.out
$ diff myprogram.out molecularmass.out
```

The first line executes the `Java` program `MolecularMass` by redirecting input from the file `molecularmass.in` and redirecting output to the file `myprogram.out`. The second command performs a file difference operation between your output file `myprogram.out` and the intended output file `molecularmass.out`. If these files match exactly, then the output of the `diff` command displays nothing. Otherwise, it lists all the lines that are different. Please make sure as you test your programs, that the output matches exactly with the intended output.

Submission: Please submit the files `IntStack.java` and `MolecularMass.java` as a single zip archive `h02.zip` through ASULearn. The zip archive should only contain the individual files of the assignment, and these files should not be inside folders. Moreover, please do not include other extraneous files. Only include all files that belong to your solution.

Input/ Output Instructions: For all programs, until and otherwise stated, we will be taking the input from standard input (`System.in`) and will be sending the output to standard output (`System.out`). Since we will be using an automatic grading system, please make sure that your output format exactly matches the sample outputs. So make sure that there are no extraneous output in terms of spaces, newlines and other characters.

Notes on Coding: Please do not include user-defined packages in your code. Your code should run in the Unix/Linux machine using the commands `javac` and `java`.

Please note that you are **not allowed to use Java Collections** which contain pre-defined libraries for many of the data structures that we learn in class. The purpose of these assignments is to build these data structures from first principles. Programs that includes these objects will receive 0 points.