COMPSCI 2S03, Principles of Programming
Assignment 3, Fall 2019

Hassan Ashtiani, McMaster University

Due date: Monday, October 21, 9pm

- You should write your codes in Java. The input/output of the programs are all from/to standard-input/standard-output (i.e., reading from the keyboard, and writing to the text terminal).
- DOMJudge is used to automatically test your code against a number of testcases. To submit to DOMJudge, login with your username and password on http://130.113.70.122/domjudge/team from the campus (or via VPN) and then submit your source files for each question.
- In addition to DOMJudge, submit your source codes on Avenue. Upload a single zip file named `macID_assignment3.zip` (e.g., `adtdb12_assignment3.zip`). The zip file should include only one folder, called `adtdb12_assignment3`, and this folder should include all your .java source files (and nothing else). This makes marking much faster for the TAs, so **any violations from this will be penalized**.
- Your source codes will be graded by the TAs; so follow the style conventions for coding from the class (naming, indentation, spacing, comments) and good coding practise to get full marks.
- Use Piazza for discussions and clarifications about the assignment.
- This assignment has bonus points. If you get a grade above 100, then it can compensate for your other assignments (but not for the exams).

1. [**80 points: 30 for testcases, 40 for source code, 10 for styling**]

   We define the @ symbol to be the operator that takes the "minimum" of two integers. For example, the result of (2@34) is 2. Furthermore, we define the & operator to be the maximum of two integers. For example, the result of (45&49) is 49. We also allow for the usual use of parentheses to show the order of applying operators. For example, the expression (22@((12@45)&14)) is equivalent to (22@(12&14)) and therefore also equivalent to (22@14) and finally equivalent to 14. So we start applying the operators from the innermost parentheses. In this question we want to write a program that takes an expression of the above form and outputs the final result (which is an integer).

Input Sample	Output Sample
`22@((12@45)&14)`	`14`

   Here is another example.

Input Sample	Output Sample
`(((22@(55&(55)))&(33&44))&((40@23)&(50@25)))`	`44`

In this question you can assume that the input is **valid**, i.e., follows all the following assumptions.

- The whole input is given in a single line. There are no white-spaces between the characters. The set of all valid characters that can be used in a line includes only digits, &, @, (, and). All the numbers are non-negative, so there are no '-' characters in the input.
- All the parentheses are validly matched. For example, we do not have)2@3(nor (2@3(nor (2&4)) as the input. Also, the operators and operands are appropriately matched so the input cannot be 2@@3 or 3&.
- The length of the input is at most 1000 characters.
- The input can have additional pairs of parentheses. For example (30@34), ((30@34)) and (((30@34))) are all valid. Also, 34, (34), and 22&(33&(3)) are all valid inputs.
- We can have a series of more than one consecutive operations. For example, 2@3@4 and (3&65&33&66) are valid inputs.
- When we have more than one consecutive operations we would evaluate the expression from left to right. For example, consider (3&(2@3)&33@66): this is equivalent to (3&2&33@66) by just replacing the innermost parentheses. After that, we have 3 operations in the same level and therefore we begin from the left (by evaluating 3&2 which is 3) and we would have (3&33@66). Next we will have (33@66). Finally, the last operation's outcome is 33.

2. [**40 points: 20 for testcases, 20 for source code**]

   In the previous question we assumed that the input is always valid. Here, we want to build on the previous question but add the following exception handling mechanisms to our code:

   - The input can have invalid characters (as defined in the previous question). In this case, the program should output "INVALID CHARACTERS".
   - If there are no invalid characters, but the input is still invalid (as defined in the previous question), the program should output "INVALID EXPRESSION"
   - If the input is valid, the program should run normally and output the result (as in question 1).

   Here is an example with whitespace (which is an invalid character).

Input Sample	Output Sample
`(22@(  (12@45)&14))`	`INVALID CHARACTERS`

Here is another example.

Input Sample	Output Sample
`(−22@((12@45)&14))`	`INVALID CHARACTERS`

And yet another one.

Input Sample	Output Sample
`(22+((12@45)14z))`	`INVALID CHARACTERS`

Here is an examples for mismatch between the parentheses.

Input Sample	Output Sample
`(((22@(55&55))&(33&44))&(((40@23)&(50@25)))`	INVALID EXPRESSION

The following one has both invalid characters and invalid parentheses...but as mentioned above, the program should output INVALID CHARACTERS.

Input Sample	Output Sample
`(22@((12@45)&14)) (`	INVALID CHARACTERS