Programming Assignment

Just “Stack” It Next to Me

Overview. In Assignment 2, you used the Array class to design and implement a Stack. Likewise, we have covered the algorithm for converting an Infix expression to a Postfix expression in class. The motivation for this is it allows the expression to be processed in O(n) time.

Requirements. You will complete the first implementation of the basic expression evaluator using the Stack created in Assignment 2 and the Infix to Postfix algorithm discussed in class. Moreover, to improve the design of your expression evaluator, you will use the Command Pattern (as explained in class) to evaluate the Postfix expression, and the (Abstract) Factory to create the commands based on parsing the Infix expression to convert it to a Postfix expression.

Each new class added for this assignment must be implemented in its own file.

You must create a complete program (i.e., your program must have a main function that runs). The expression evaluator must be able to handle the following operators/tokens: +, -, /, *, %, (,), integers (positive and negative), and variables. If the expression has variables, then you must prompt for the value of each variable in alphabetical order. All expression will have a space between each token to simplify parsing. All input should come from the STDIN. The program must loop until the user types QUIT—notice the all caps. All output should go to STDOUT. Failure to follow the required input/output method will result in lose in all execution points. All assignments must run on Telsa.

The name of the generate executable must use the **exename** property in the .mpc file to generate an executable. The name of your executable does not matter as the grading system will learn how to execute our program from the project specification.

Assignment files. No files will be provided for this assignment. You are to update your MPC file accordingly so that you can build your expression evaluator. The name of the MPC file must be **assignment3.mpc**.

Submissions. All submissions must be placed on IU Github by the deadline. The Github repository must be **private** under your account and named

cs36300-spring2018-calculator1

Failure to upload all your source code and project files to a Github repo with this name will receive an automatic zero (0). You must add the TA and me (hilljh) as collaborators to the project. Failure to add the TA or me as a collaborator will result in an automatic zero (0).

Example Program Execution

Below is an example program execution.

```
Please enter your expression: x + 9
x = 5
The answer is: 14
Please enter your expression: 9 * 5
The answer is: 45
Please enter your expression: 25 * ( y + 1 )
y = 1
The answer is: 50
QUIT
```