SE 461 – Spring 2021
Assignment #3- Stacking Patterns
Due: 3/23/2021

In Assignment 2, you used the Array class to design and implement a Stack through the use of system composition and C++ templates. We want to build upon this implementation in order to build a simple expression evaluator, calculator if you will. In lecture, we will cover the algorithm for converting an Infix expression to a Postfix expression. The motivation behind this conversion was that it allows us to process it in `O(n)` time.

For this assignment, 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 lecture. Moreover, to improve the design of your expression evaluator, you will need to use the **Command Pattern** to evaluate the Postfix expression, and the **(Abstract) Factory Pattern** to create the commands based on parsing the Infix expression to convert it to a Postfix expression.

Development Process:
For this assignment, all development must take place on the **master branch**. It is required that you commit and push often! Please ensure that your comments on your commits are meaningful and appropriate.

You are to create a complete working program in this assignment. The expression evaluator **must** be able to handle the following operators/tokens: **`+`**, **`-`**, **`/`**, **`*`**, `%`, **`(`**, **`)`**, **`integers`** (both positive and negative). 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`.

Note: Failure to follow the required input/output method will result in an automatic 40-point deduction.

Submission:
All assignments must be submitted on Butler GitHub (github.butler.edu). The name of your Butler GitHub repository must be as follows: **se461_spring2021_stacking**. I must be added as collaborator to your Private repository.

For this assignment I am not providing you a list of files that are required – you will need to decide this on your own. Obviously, you could accomplish this in a single class file – but as we know that would not be a good design choice. Please make use of the concepts that we have covered. You are expected to submit any and all source/header/template files that are required to run your program – you are also required to submit an MPC and Valgrind.txt file with your submission on GitHub. You are required to include **`driver.cpp`** as a Source File in your MPC file.