Project 4

Stack Application

CS 240 - Section 1

Kristin Adachi

knadachi@cpp.edu

February 12, 2016

Section 1: Project Specification

This project focused on the implementation of stacks and applying it to the conversion of arithmetic expressions. We were required to write two methods that converted a user given infix expression to postfix and prefix expressions. The program also needed to continue asking the user for infix expressions until they entered nothing to exit. I was able to meet this requirement by implementing my own stack composed of a singly linked list (the head of the singly linked list was the top of the stack). I also created two methods called `postfix` and `prefix` that converts an infix expression to postfix and prefix expressions respectively. Rather than organizing this into one class, I separated the conversion functions into one class and the user interface in another. Due to this approach, my resulting code is much more organized and cleaner than it would have been if I hadn't.

Section 2: Testing Methodology

My program allows the user to input any infix expression and converts it to postfix and prefix expressions if able to. My `validate` method checks if the user's infix expression is convertible. For instance, it checks for mismatched parentheses and for the right format of an infix expression (certain amount of operands and operators as well as their order).

To test the effectiveness of my code, I wrote down several, valid cases and solved them each by hand. The first seven inputs in my output text file are the valid cases I described. I tested some cases with parentheses and others without. I also tested for the precedence of the operations by rearranging the operators and comparing the converted expressions to the ones I found by hand. As a result, all of the infix expressions I tested were converted accurately and successfully.

I also tested cases of inputs that should return a message saying that the user inputted an expression that could not be converted. The last four inputs in my output text file are the invalid cases I described. To do so, I tested cases where two operators were in a row and also when two operands were in a row. Then, I tested for mismatched parentheses and parentheses that were out of order. Again, my program was successful because it displayed the "Invalid Input" statement for each of these cases.

Throughout the testing process, I ran into several issues such as incorrect outputs. It occurred with both my `postfix` and `prefix` conversion methods. However, I was able to fix it after reconsidering the logic behind my code and editing conditions. I ran into more problems when I was writing and testing my validate method. At first, it wouldn't catch all the invalid input cases and ran the conversion methods for invalid expressions. Again, I needed to edit my conditions in order to fix the issues.

Section 3: Lessons Learned

Throughout the process of writing my program, I was able to become more familiar with implementing a stack. It also helped me reinforce my knowledge on singly linked lists because that was the data structure I chose to use for my stack. I chose to use a singly linked list instead of an array because we wouldn't know the exact size we needed for our stack.

I implemented the `postfix` method as the directions explained, but I had trouble picturing how the `prefix` method worked. As a result, I ended up using a different approach to convert the infix expression to a prefix expression. Rather than using two stacks, I only used one for the operators. I reversed the infix string, used the postfix method with edited conditions, and then reversed the final string again. As a result, it produced an accurate prefix expression.

Overall, the project ended up being more difficult than I had expected. Though it seemed simple and straightforward at first, I struggled when I began implementing the `prefix` method. What really helped me throughout the process was simply drawing stacks and going through my program line by line. Though this project wasn't as difficult as the hash table project, I believe it still provided a good challenge for those unfamiliar with the implementation of stacks.