Nicholas Ang
CS302

# Homework 1 Documentation

## StackArray Implementation:

The StackArray implementation was done based on the specifications given to us. Given that I had Stack.h as a template, I created StackArray.cpp to implement the functions in StackArray.h. The methods in StackArray.cpp were the default constructor, copy constructor, assignment operator overload, destructor, push, pop, isEmpty, isFull, clear, and showStructure. The showStructure function was given to us in show6.cpp so I did not implement another one. The rest of the functions, however, were implemented by me. The push and pop functions used exceptions to produce an error message when something did not work such as being able to push when the stack was full or pop when the stack was empty. To test out the array stack, I changed the value in config.h to 0 and compiled test6.cpp with cmake.

## StackLinked Implementation:

The StackLinked implementation was similarly done as StackArray. Given that I had Stack.h as a template, I created StackLinked.cpp to implement the functions in StackLinked.h. The methods in StackLinked.cpp were the default constructor, copy constructor, assignment operator overload, destructor, push, pop, isEmpty, isFull, clear, and showStructure. Additionally I had to declare the StackNode constructor and implement it to be able to create new nodes to the list. The showStructure function was given to us in show6.cpp so I did not implement another one. The rest of the functions were implemented by me. The push and pop functions used exceptions to produce an error message when something did not work such as being able to push when the stack was full or pop when the stack was empty. The exception for push would never occur because the linked list stack would never reach a “full” capacity. To test out the linked list stack, I changed the value in config.h to 1 and compiled test6.cpp with cmake.

## Postfix Implementation:

The postfix evaluation program is able to evaluate postfix expressions. In the main function is the menu which has three options. The first option displays the 5 test cases that display their infix and postfix notations before being evaluated. The second option allows the user to input their own postfix expression. However, the postfix expression has to be valid to be evaluated properly. It will evaluate incorrectly if part of the expression is missing. The postfix evaluation function allows for addition, subtraction, multiplication, division, and exponents to be evaluated in postfix notation. It uses the StackLinked implementation to pop values for evaluation and push calculated values back onto the stack until it fully computes the expression. I created the powerPos function to be able to deal with and calculate positive exponents. It

recursively multiplies value a by itself b times until it reaches the exponent value of 0. The third option exits the program. To test the postfix program, I compiled with cmake and ran the executable.

DelimitersOk implementation:

The delimitersOk program checks a string for matching delimiters to test whether the expression is valid or invalid. In the main function, there is a menu that has three options. The first option displays the 5 test cases that are checked with the delimitersOk function and outputs their validity. The second option allows the user to input their own string to check for matching and valid delimiters. The third option exits the program. The delimitersOk function uses the linked list stack to find matching delimiters. When a left side delimiter is found, it is pushed onto the stack. When a right side delimiter is found, it checks if the popped value from the stack is equal to the left side delimiter. If the delimiter does not match or is by itself, the expression is invalid. Since delimitersOk is a bool function, the program outputs “Valid” when it is true and “Invalid” when it is false. To test the delimiters program, I compiled with cmake and ran the executable.

I compiled the test6 executable, the postfix executable, and the delimiters executable with cmake. In the folder with the CMakeLists.txt file, I entered cmake ./ into the terminal. Then I entered make all to create all the executables. For the test6 program, it will have to be recompiled when changing the value in config.h from 0 and 1.