**CSE 330 Lab 1 Report**

Daniel Meyer

Data Structures

Fall 2017

**Status:** 100%

**Time Complexity:** O(n)

**Storage Complexity:** O(1)

**Source Code: \***See attached

**Sample Run:** \*See attached

**Source Code**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Daniel Meyer

\*expr.cpp

\*9/25/17

\*Lab 1: Infix-to-Postfix Expression Conversion

\*Implement an algorithm that converts infix expressions to postfix expressions.

\*CSE 330

\*Fall 2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stack>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

/\*

Function to test precedence of each operand in the stack

to determine if operands to to be popped off or stored.

\*/

int prec(char c)

{

switch (c){

case '(': return 0;

case '\*':

case '/': return 1;

case '+':

case '-': return 2;

default: return 3;

}

}

/\*

Main function that implements infix to postfix expression conversion.

\*/

int main()

{

stack<char> operators;

char next\_character;

ifstream inFile("test.txt"); //allows for more versatility vs single input

if (!inFile) { //checks for file existence

cout << "Error reading file" << endl;

}

//cin >> next\_character; //single caracter style input

//while (!cin.eof()) {

while(inFile.get(next\_character)) {

if (next\_character >= 'A' && next\_character <= 'Z' || next\_character >= 'a' &&

next\_character <= 'z' || next\_character >= '0' && next\_character <= '9') {

cout << next\_character;

}

else {

if (next\_character == '(') {

operators.push(next\_character);

}

else if (next\_character == ')') {

while (!operators.empty() && operators.top() != '(') {

cout << operators.top();

operators.pop();

}

if (!operators.empty()) {

operators.pop();

}

else {

cout << "Error" << endl;

}

}

else if (next\_character == '\*' || next\_character == '/' ||

next\_character == '+' || next\_character == '-') {

if (operators.empty() || prec(operators.top()) < prec(next\_character))

operators.push(next\_character);

else {

while (!operators.empty() && prec(operators.top())

>= prec(next\_character)) {

cout << operators.top();

operators.pop();

}

operators.push(next\_character);

}

}

else {

cout << "Error" << endl;

}

}

}

while (!operators.empty()) {

cout << operators.top();

operators.pop();

}

cout << endl; //Cleans up final output

inFile.close();

return 0;

}

**Sample Run**

