Data Structures and Algorithms Lab Journal - Lab 6

Name: SYED MURTAZA HAIDER

Enrollment #: 01-134232-182

Class/Section: BS CS-3D

**Objective**

This lab session is aimed at introducing students to Stack and its applications.

**Task 1 :**

Give answers to the following.

|  |  |
| --- | --- |
|  | A |
|  | B A |
|  | C B A |
|  | B A |
|  | A |
|  | D A |
|  | E D A |
|  | D A |
|  | A |

|  |  |
| --- | --- |
| 1. | Show the contents of stack (at each step) once the following sequence of statements is executed.  Stack S;   1. S.push(‘A’); 2. S.push(‘B’); 3. S.push(‘C’); 4. S.Pop(); 5. S.Pop(); 6. S.push(‘D’); 7. S.push(‘E’); 8. S.Pop(); 9. S.Pop(); |
| 2. | Convert (manually) the following expressions to postfix.  **(A+B\*D)/(E-F)+G : A B D \* + E F - / G +** |

|  |  |
| --- | --- |
| **Symbol** | **Stack Contents** |
| **2** |  |
| **7** |  |
| **3** |  |
| **-** |  |
| **/** |  |
| **2** |  |
| **1** |  |
| **5** |  |
| **+** |  |
| **\*** |  |
| **+** |  |

|  |  |
| --- | --- |
|  |  |
|  | 2 |
|  | 7,2 |
|  | 3,7,2 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  | **A+(B-D)/E-F\*(G\*H+K) : A B D \* + E F - / G H \* K + \* -A** |
| 3. | Evaluate the given Postfix expression and trace the contents of the Stack at each step using the standard evaluation algorithm.  **2 7 3 - / 2 1 5 + \* +**  **Result : 23/2** |

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Stack Contents** | **Output Expression** |
| **(** | **(** |  |
| **A** | **(** | A |
| **+** | **+** |  |
| **B** | **+** |  |
| **)** |  |  |
| **\*** | **\*** |  |
| **C** | **\*** |  |
| **-** | **-** |  |
| **D** | **-** |  |
| **+** | **+** |  |
| **F** | **+** |  |
| **\*** | **+** |  |
| **G** |  |  |

**Task 2 :**

Convert the following expression from infix to postfix and show the contents of Stack and the output expression at each step.

(A+B) \* C – D+F\*G

4.

Implement the Stack class and employ it to solve the given exercises.

**Exercise 1**

Write a C++ program that prompts user to enter a number. Define two functions (a) void dectobin(int) and (b) void dectooct(int). Each function should take the decimal number entered

by the user as an argument and print the (a) binary (b) octal equivalent of the number respectively using stack.

**// Function to convert decimal to binary using a stack**

**void dectobin(int num) {**

**Stack stack(32);**

**if (num == 0) {**

**cout << "Binary: 0" << endl;**

**return;**

**}**

**while (num > 0) {**

**stack.push(num % 2);**

**num /= 2;**

**}**

**cout << "Binary: ";**

**while (!stack.isEmpty()) {**

**cout << stack.pop();**

**}**

**cout << endl;**

**}**

**// Function to convert decimal to octal using a stack**

**void dectooct(int num) {**

**Stack stack(32);**

**if (num == 0) {**

**cout << "Octal: 0" << endl;**

**return;**

**}**

**while (num > 0) {**

**stack.push(num % 8);**

**num /= 8;**

**}**

**cout << "Octal: ";**

**while (!stack.isEmpty()) {**

**cout << stack.pop();**

**}**

**cout << endl;**

**}**

**Exercise 2**

Write a program that reads a string (an array of characters) from a text file. Reverse the string using the Stack and write the reversed string to another text file.

**#include <iostream>**

**#include <fstream>**

**#include <stack>**

**using namespace std;**

**void reverseStringUsingStack(const string& inputFile, const string& outputFile) {**

**ifstream inFile(inputFile);**

**ofstream outFile(outputFile);**

**// Check if the input file opened successfully**

**if (!inFile) {**

**cerr << "Error opening input file." << endl;**

**return;**

**}**

**// Check if the output file opened successfully**

**if (!outFile) {**

**cerr << "Error opening output file." << endl;**

**return;**

**}**

**string str;**

**getline(inFile, str); // Read the entire line from the file into a string**

**// Stack to hold characters for reversing**

**stack<char> charStack;**

**// Push all characters onto the stack**

**for (char ch : str) {**

**charStack.push(ch);**

**}**

**// Pop characters from stack to reverse the string**

**string reversedStr;**

**while (!charStack.empty()) {**

**reversedStr += charStack.top();**

**charStack.pop();**

**}**

**// Write the reversed string to the output file**

**outFile << reversedStr;**

**// Close files**

**inFile.close();**

**outFile.close();**

**cout << "Reversed string written to " << outputFile << endl;**

**}**

**int main() {**

**string inputFile = "input.txt";**

**string outputFile = "output.txt";**

**reverseStringUsingStack(inputFile, outputFile);**

**return 0;**

**}**

**Exercise 3**

Write a function that reads a Mathematical expression from a text file and verifies the validity of paranthesis in the expression using a Stack.

**#include <iostream>**

**#include <fstream>**

**#include <stack>**

**#include <string>**

**using namespace std;**

**bool areParenthesesBalanced(const string& expression) {**

**stack<char> s;**

**for (char ch : expression) {**

**// Push opening brackets onto the stack**

**if (ch == '(' || ch == '{' || ch == '[') {**

**s.push(ch);**

**}**

**// Check for closing brackets**

**else if (ch == ')' || ch == '}' || ch == ']') {**

**// If stack is empty or top doesn't match, parentheses are unbalanced**

**if (s.empty()) {**

**return false;**

**}**

**char top = s.top();**

**s.pop();**

**if ((ch == ')' && top != '(') ||**

**(ch == '}' && top != '{') ||**

**(ch == ']' && top != '[')) {**

**return false;**

**}**

**}**

**}**

**// Stack should be empty if all parentheses were matched**

**return s.empty();**

**}**

**void checkParenthesesInFile(const string& filename) {**

**ifstream inFile(filename);**

**if (!inFile) {**

**cerr << "Error opening file." << endl;**

**return;**

**}**

**string expression;**

**getline(inFile, expression);**

**if (areParenthesesBalanced(expression)) {**

**cout << "The parentheses in the expression are balanced." << endl;**

**} else {**

**cout << "The parentheses in the expression are NOT balanced." << endl;**

**}**

**inFile.close();**

**}**

**int main() {**

**string filename = "expression.txt";**

**checkParenthesesInFile(filename);**

**return 0;**

**}**

**Exercise 4**

Implement a program to read a postfix expression from a text file, evaluate the expression using a Stack and display the result. The text file should contain expressions in the form as illustrated in the following. (For simplicitiy, assume single digit numbers in the expression.)

23+5\*6+

**#include <iostream>**

**#include <fstream>**

**#include <stack>**

**#include <cctype>**

**using namespace std;**

**// Function to evaluate a postfix expression**

**int evaluatePostfix(const string& expression) {**

**stack<int> s;**

**for (char ch : expression) {**

**if (isdigit(ch)) {**

**// Convert character to integer and push onto stack**

**s.push(ch - '0');**

**} else {**

**// Check for valid operators and pop two operands**

**if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {**

**int operand2 = s.top(); s.pop();**

**int operand1 = s.top(); s.pop();**

**// Perform the operation and push the result back onto the stack**

**int result;**

**switch (ch) {**

**case '+': result = operand1 + operand2; break;**

**case '-': result = operand1 - operand2; break;**

**case '\*': result = operand1 \* operand2; break;**

**case '/': result = operand1 / operand2; break;**

**}**

**s.push(result);**

**}**

**}**

**}**

**// The final result will be at the top of the stack**

**return s.top();**

**}**

**// Function to read postfix expression from a file and evaluate it**

**void evaluatePostfixFromFile(const string& filename) {**

**ifstream inFile(filename);**

**if (!inFile) {**

**cerr << "Error opening file." << endl;**

**return;**

**}**

**string expression;**

**getline(inFile, expression);**

**int result = evaluatePostfix(expression);**

**cout << "The result of the postfix expression is: " << result << endl;**

**inFile.close();**

**}**

**int main() {**

**string filename = "postfix\_expression.txt";**

**evaluatePostfixFromFile(filename);**

**return 0;**

}

**Implement the given exercises and get them checked by your instructor.**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Exercise** | **Checked By:** |
| 1. | Exercise 1 |  |
| 2. | Exercise 2 |  |
| 3. | Exercise 3 |  |
| 4. | Exercise 4 |  |

++++++++++++++++++++++++++++++++++++