LAB # 05 - ASSIGNMENT

20

NOV

2023

**GROUP 16**

MUHAMMAD YASIR | CT-22082

MUHAMMAD SHAHEER QURESHI | CT-22090

SYED SAAD WAQAR | CT-22097

AMMAR YASSER AHMED | CT-22103

NOFIL AHMED KHAN | CT-22301

DATA STRUCTURES ALGORITHMS AND APPLICATIONS (ct – 159)

TEACHER : SIR ABDUL KARIM KAZI

**EXERCISE**

QUESTION 1:

/\*

DSA Lab 5

Group 16

Question 1

1. Please write a program which performs the following tasks:

a. Make a left to right scan of the postfix expression

b. If the element is an operand push it on Stack

c. If the element is operator, evaluate it using as operands the correct number from stack

and pushing the result onto the stack

\*/

#include <iostream>

#include <cstdlib>

using namespace std;

class Stack {

private:

static const int MAX\_SIZE = 100;

int arr[MAX\_SIZE];

int top;

public:

Stack() {

top = -1;

}

bool isEmpty() {

return top == -1;

}

void push(int value) {

if (top >= MAX\_SIZE - 1) {

cout << "Stack Overflow\n";

return;

}

arr[++top] = value;

}

int pop() {

if (isEmpty()) {

cout << "Stack Underflow\n";

return -1;

}

return arr[top--];

}

int peek() {

if (isEmpty()) {

cout << "Stack is empty\n";

return -1;

}

return arr[top];

}

};

int evaluatePostfix(const string expression) {

Stack operandStack;

int i = 0;

char c;

for (c = expression[i]; c != '\0'; i++) {

c = expression[i];

if (c >= '0' && c <= '9') {

operandStack.push(c - '0'); // Push operands onto the stack

} else if (c == '+' || c == '-' || c == '\*' || c == '/') {

int operand2 = operandStack.pop();

int operand1 = operandStack.pop();

switch (c) {

case '+':

operandStack.push(operand1 + operand2);

break;

case '-':

operandStack.push(operand1 - operand2);

break;

case '\*':

operandStack.push(operand1 \* operand2);

break;

case '/':

operandStack.push(operand1 / operand2);

break;

}

}

}

return operandStack.peek();

}

int main() {

string postfixExpression = "122/+45\*-"; // Change this to your desired postfix expression

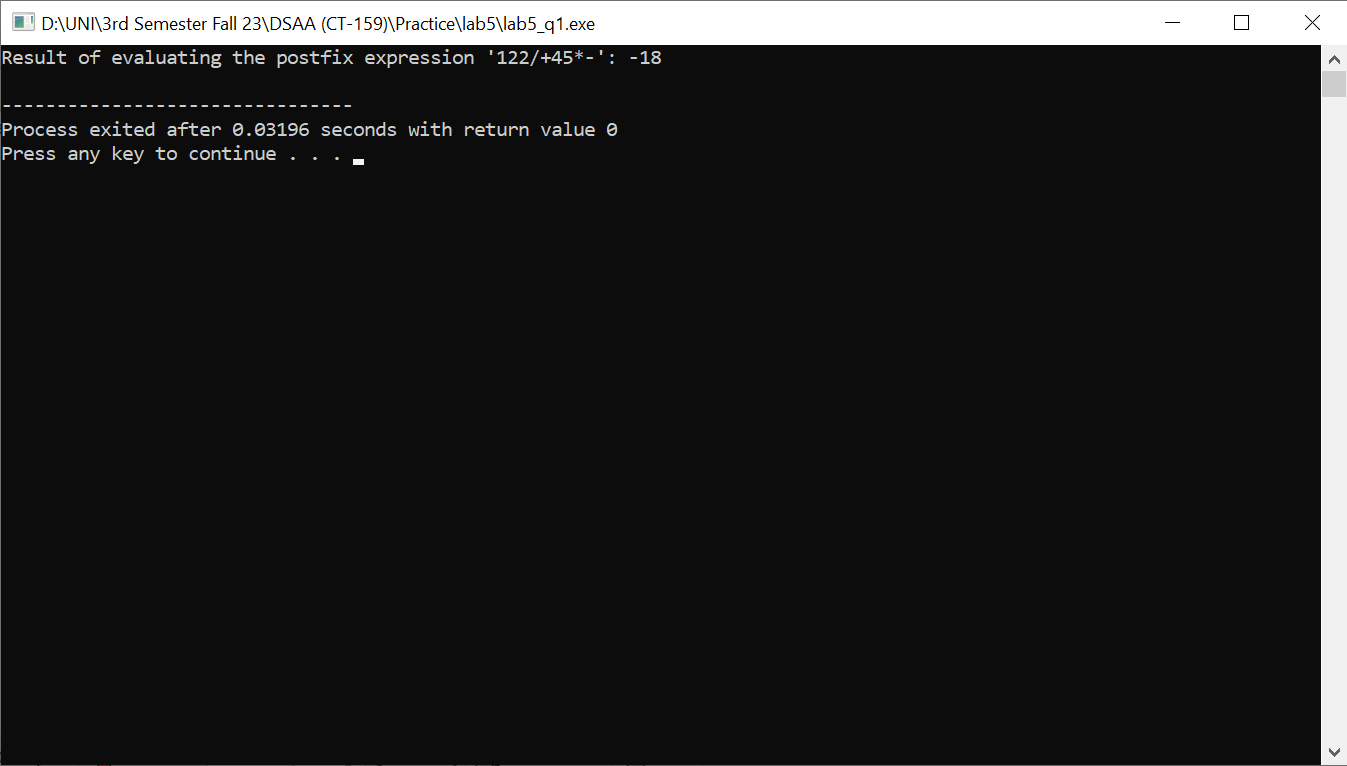
int result = evaluatePostfix(postfixExpression);

cout << "Result of evaluating the postfix expression '" << postfixExpression << "': " << result << endl;

return 0;

}

OUTPUT:



QUESTION 2:

/\*

DSA LAB 5

Group 16

Question 2

2. A palindrome is a word, phrase, number, or another sequence of characters that reads the same

backward and forwards. Can you determine if a given string, s, is a palindrome? Write a

Program using stack for checking whether a string is palindrome or not.

\*/

#include<iostream>

#include<cstdlib>

using namespace std;

struct Node{

int data;

struct Node\* next;

};

class Stack{

Node \*top;

public:

Stack(){

top = NULL;

}

bool isEmpty(){

return top == NULL;

}

bool isFull(){

Node\* temp = (struct Node\*)malloc(sizeof(struct Node));

return temp == NULL;

}

void push(int data){

if(isFull()){

cout << "Stack Overflow" << endl;

return;

}

Node \*temp = (struct Node\*)malloc(sizeof(struct Node));

temp->data = data;

temp->next = top;

top = temp;

}

int pop(){

if(isEmpty()){

cout << "Stack Underflow" << endl;

return -1;

}

Node\* temp = top;

top = top->next;

int value = temp->data;

free(temp);

return value;

}

int peek(int position){

Node\* temp = top;

int i = 0;

for(; ((temp != NULL) && (i != position)); i++){

temp = temp->next;

}

if(temp != NULL){

return temp->data;

}

return -1;

}

int stackTop(){

return top->data;

}

int stackBottom(){

Node\* temp = top;

while(temp->next != NULL){

temp = temp->next;

}

return temp->data;

}

Stack palindrome(){

Stack temp;

Node \*tempNode = top;

int i;

for(i = 0; ;i++){

if(tempNode == NULL){

break;

}

temp.push(peek(i));

tempNode = tempNode->next;

}

i--;

Stack temp2;

while(1){

if(i == -1){

break;

}

temp2.push(temp.peek(i));

i--;

}

return temp2;

}

bool checkPalindrome(Stack S1, Stack S2){

bool correct = true;

Node\* temp = top;

for(int i = 0; temp != NULL; i++){

temp = temp->next;

if(S1.peek(i) == S2.peek(i)){

correct = true;

}

else{

return false;

}

}

return correct;

}

void traversal(){

cout << "Stack Traversal: ";

Node\* temp = top;

while(temp != NULL){

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

};

int main(){

Stack S1;

S1.push(1);

S1.push(2);

S1.push(3);

S1.push(3);

S1.push(2);

S1.push(1);

cout << "S1: ";

S1.traversal();

Stack S2 = S1.palindrome();

cout << "S2: ";

S2.traversal();

if(S1.checkPalindrome(S1, S2))

cout << "S1 and S2 are palindrome" << endl;

else

cout << "S1 and S2 are not palindrome" << endl;

Stack S3;

S3.push(1);

S3.push(2);

S3.push(3);

S3.push(4);

S3.push(5);

S3.push(6);

cout << "S3: ";

S3.traversal();

Stack S4 = S3.palindrome();

cout << "S4: ";

S4.traversal();

if(S3.checkPalindrome(S3, S4))

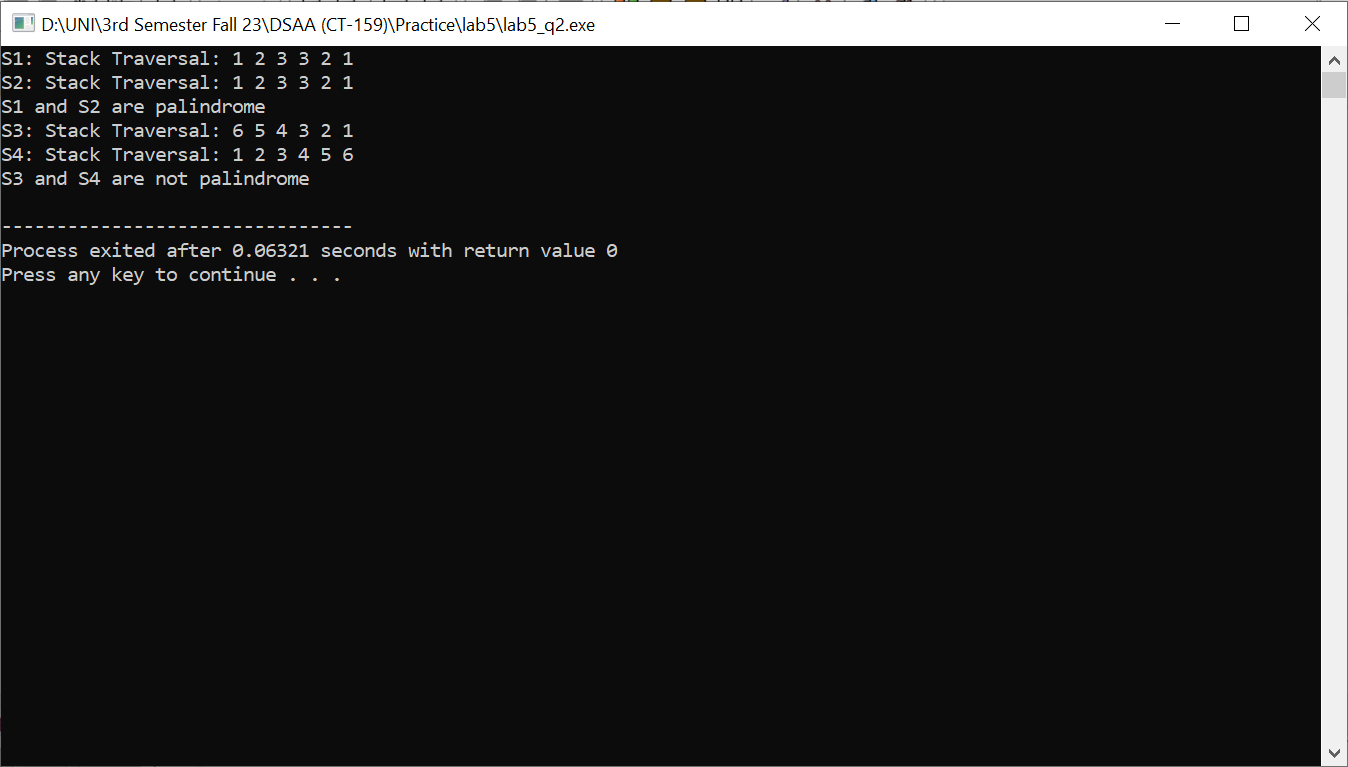
cout << "S3 and S4 are palindrome" << endl;

else

cout << "S3 and S4 are not palindrome" << endl;

}

OUTPUT:



QUESTION 3:

/\*

DSA LAB 5

Group 16

Quetion 3

3. Write a program using stacks which takes an expression as input and determines whether the

delimiters are matched or not.

\*/

#include <iostream>

#include <cstdlib>

using namespace std;

struct Node{

char data;

struct Node\* next;

};

class Stack{

Node\* top;

int size; // stores the current number of elements

int capacity; // total capacity

public:

Stack(int c = 0){

top = NULL;

size = 0;

capacity = c;

}

bool isEmpty(){

return top == NULL;

}

bool isFull(){

return size == capacity;

}

int getSize() const{

return size;

}

int getCapacity() const{

return capacity;

}

void push(char data){

if(isFull()){

cout << "Stack Overflow" << endl;

return;

}

Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = top;

top = newNode;

size++;

}

void pop(){

if(isEmpty()){

cout << "Stack underflow" << endl;

return;

}

Node\* temp = top;

top = top->next;

free(temp);

size--;

}

void setCapacity(int c){

capacity = c;

}

char stackTop(){

return top->data;

}

char stackBottom(){

Node\* temp = top;

while(temp->next != NULL){

temp = temp->next;

}

return temp->data;

}

char peek(int position){

Node\* temp = top;

for(int i = 0; ((temp != NULL) && (i != position)); i++){

temp = temp->next;

}

if(temp != NULL){

return temp->data;

}

return -1;

}

};

bool delimitermatch(string s, Stack st)

{

int size = s.size();

for (int i = 0; i < size; i++)

{

if (s[i] == '{' || s[i] == '(' || s[i] == '[')

{

st.push(s[i]);

continue;

}

else if (s[i] == '}' || s[i] == ')' || s[i] == ']')

{

if (st.isEmpty())

{

return false; // More closing delimiters than opening delimiters

}

if (st.stackTop() == '(' && s[i] == ')')

{

st.pop();

}

else if (st.stackTop() == '[' && s[i] == ']')

{

st.pop();

}

else if (st.stackTop() == '{' && s[i] == '}')

{

st.pop();

}

else

{

return false;

}

}

}

if (st.isEmpty())

{

return true;

}

else

{

return false;

}

}

int main()

{

Stack st(33);

string s = "{}()()[]{([dsjhkjshdhsajdghfhf])}";

cout << "String is: " << s << "\nSize: " << s.size() << endl;

int n = delimitermatch(s, st);

if (n == 1)

{

cout << "Delimiters matched";

}

else

{

cout << "Delimiters not matched";

}

}

OUTPUT:

