NAME KHAWAR WAQAS

REG# SP22-BCS-025

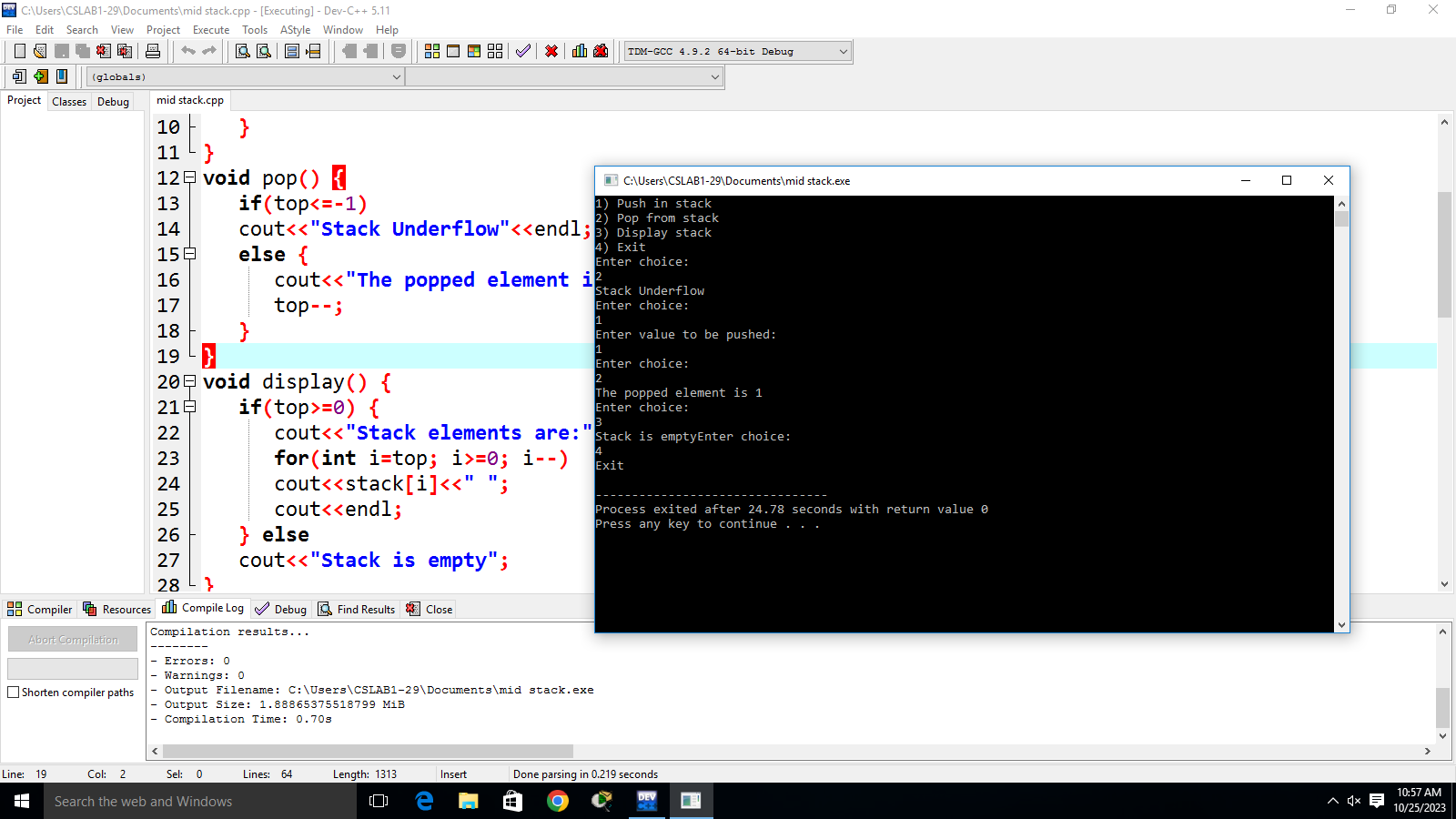
SUBJECT DSA LAB

SECTION “A”

QUESTION NO 2

Amplement stack using array

Code



#include <iostream>

using namespace std;

int stack[100], n=100, top=-1;

void push(int val) {

if(top>n-1)

cout<<"Stack Overflow"<<endl;

else {

top++;

stack[top]=val;

}

}

void pop() {

if(top<=-1)

cout<<"Stack Underflow"<<endl;

else {

cout<<"The popped element is "<< stack[top] <<endl;

top--;

}

}

void display() {

if(top>=0) {

cout<<"Stack elements are:";

for(int i=top; i>=0; i--)

cout<<stack[i]<<" ";

cout<<endl;

} else

cout<<"Stack is empty";

}

int main() {

int ch, val;

cout<<"1) Push in stack"<<endl;

cout<<"2) Pop from stack"<<endl;

cout<<"3) Display stack"<<endl;

cout<<"4) Exit"<<endl;

do {

cout<<"Enter choice: "<<endl;

cin>>ch;

switch(ch) {

case 1: {

cout<<"Enter value to be pushed:"<<endl;

cin>>val;

push(val);

break;

}

case 2: {

pop();

break;

}

case 3: {

display();

break;

}

case 4: {

cout<<"Exit"<<endl;

break;

}

default: {

cout<<"Invalid Choice"<<endl;

}

}

}while(ch!=4);

return 0;

}

Question no.1

#include <iostream>

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

this->next = NULL;

}

};

class LinkedList {

public:

Node\* head;

LinkedList() {

head = NULL;

}

void addNode(int data) {

Node\* newNode = new Node(data);

newNode->next = head;

head = newNode;

}

bool isPalindrome() {

if (head == NULL) {

return true; // An empty list is considered a palindrome.

}

// Implement a custom stack using a singly linked list

Node\* stackTop = NULL;

Node\* current = head;

int length = 0;

Node\* slow = head;

while (current) {

length++;

current = current->next;

if (length % 2 == 0) {

if (length > 2) {

Node\* temp = slow->next;

slow->next = stackTop;

stackTop = slow;

slow = temp;

}

}

}

// Step 2: Compare the second half with the elements in the stack

if (length % 2 == 0) {

slow = slow->next;

}

while (slow) {

if (slow->data != stackTop->data) {

return false;

}

slow = slow->next;

stackTop = stackTop->next;

}

return true;

}

};

int main() {

LinkedList ll;

ll.addNode(1);

ll.addNode(2);

ll.addNode(2);

ll.addNode(1);

std::cout << "Linked List: ";

Node\* current = ll.head;

while (current != NULL) {

std::cout << current->data << " -> ";

current = current->next;

}

std::cout << "NULL" << std::endl;

if (ll.isPalindrome()) {

std::cout << "The linked list is a palindrome." << std::endl;

} else {

std::cout << "The linked list is not a palindrome." << std::endl;

}

return 0;

}

