**LAB MID**

**NAME :**M UMAIR GHANI

**ROLL NO :** SP22-BCS-059

**DATE :** 25-10-2023

**SECTION :** A

**INSTRUCTOR :** MAM YASMEEN JANA

**QUESTION NO 2:** IMPLEMENT STACK USING ARRAY

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

class Stack{

private:

int top;

int arr[MAX\_SIZE];

public:

Stack() {

top = -1;

}

bool isEmpty() {

return top == -1;

}

bool isFull() {

return top == MAX\_SIZE - 1;

}

void push(int data) {

if (isFull()) {

cout << "Stack is full. Cannot push." << endl;

return;

}

arr[++top] = data;

}

void pop() {

if (isEmpty()) {

cout << "Stack is empty. Cannot pop." << endl;

return;

}

--top;

}

int peek() {

if (isEmpty()) {

cout << "Stack is empty. Cannot peek." << endl;

return -1;

}

return arr[top];

}

};

int main() {

Stack stack;

cout << "Stack operations:" << endl;

cout << "1. Push" << endl;

cout << "2. Pop" << endl;

cout << "3. Peek" << endl;

cout << "4. Is Full" << endl;

cout << "5. Is Empty" << endl;

cout << "6. Quit" << endl;

int choice, data;

do {

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter data to push: ";

cin >> data;

stack.push(data);

break;

case 2:

stack.pop();

break;

case 3:

cout << "Top element: " << stack.peek() << endl;

break;

case 4:

if (stack.isFull()) {

cout << "Stack is full." << endl;

} else {

cout << "Stack is not full." << endl;

}

break;

case 5:

if (stack.isEmpty()) {

cout << "Stack is empty." << endl;

} else {

cout << "Stack is not empty." << endl;

}

break;

case 6:

cout << "Exiting program." << endl;

break;

default:

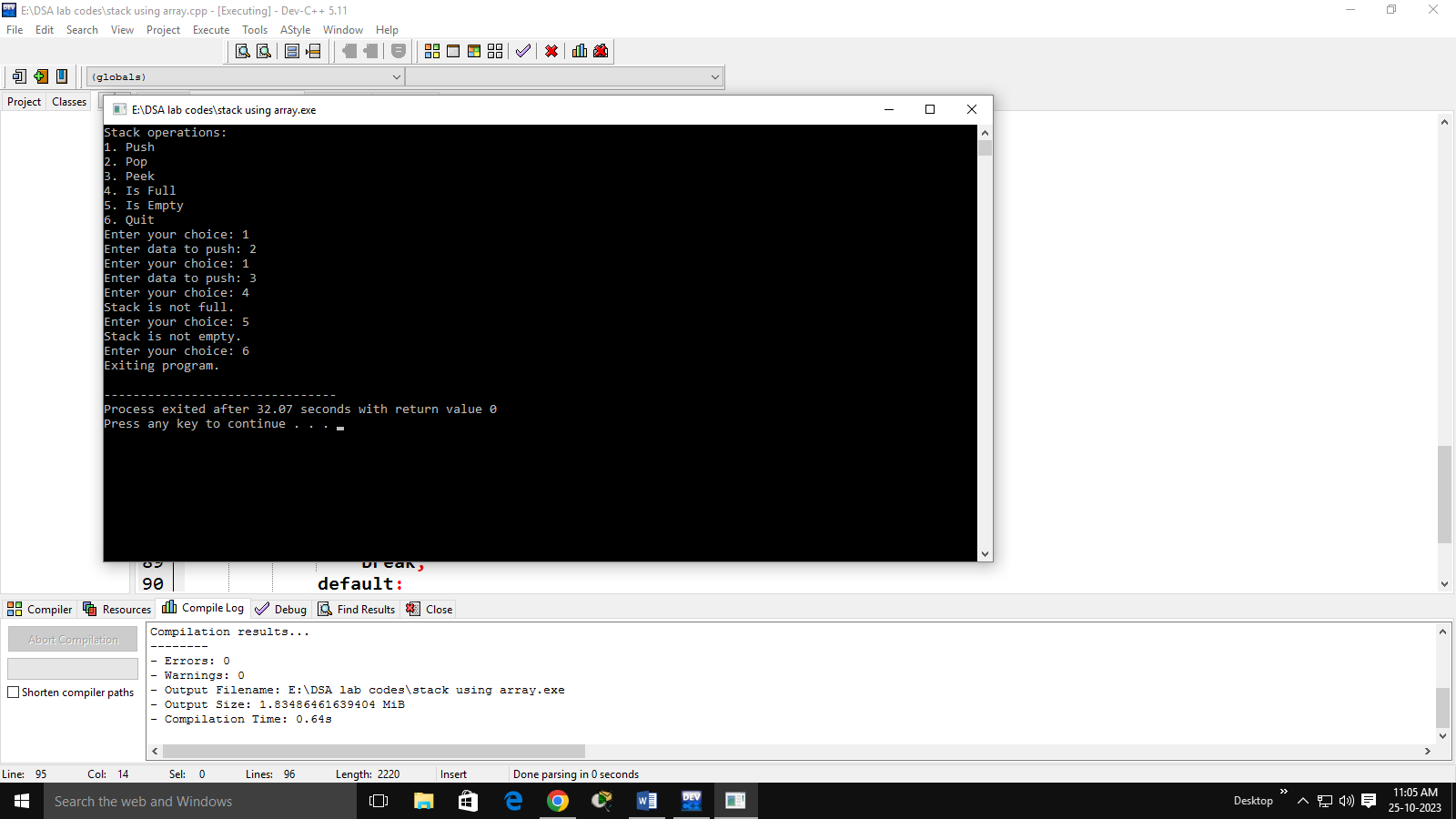
cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 6);

return 0;

}



**QUESTION NO 2:**

TO CREATE A SINGLE LINKED LIST USING SLL FUNCTION TO CHECK THAT SINGLY LINKED LIST IS PALINDRONE OR NOT.

FOR EXAMPLE 1->2->2->1

HINT: CHECK USING STACK

#include <iostream>

#include <vector>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int value){

data = value;

next = NULL;

}

};

class LinkedList {

public:

Node\* head;

LinkedList(){

head=NULL;

}

void SSL(int value) {

Node\* newNode = new Node(value);

if (!head) {

head = newNode;

} else {

Node\* current = head;

while (current->next) {

current = current->next;

}

current->next = newNode;

}

}

bool Palindrome() {

vector<int> reversedData;

Node\* current = head;

while (current) {

reversedData.insert(reversedData.begin(), current->data);

current = current->next;

}

current = head;

for (int i = 0; i < reversedData.size(); ++i) {

if (current->data != reversedData[i]) {

return false;

}

current = current->next;

}

return true;

}

};

int main() {

LinkedList myList;

myList.SSL(1);

myList.SSL(2);

myList.SSL(3);

myList.SSL(2);

myList.SSL(1);

if (myList.Palindrome()) {

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

} else {

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

}

return 0;

}

