Using a programming language of your choice, create a program that can implement a stack and all queues that we have learnt using:

Here’s a C++ program that complements a stack using both an array and a linked list:

#include <iostream>

using namespace std;

// Define a structure for the node of the linked list

struct Node {

int data;

Node\* next;

};

// Define a class for the stack implemented using an array

class ArrayStack {

private:

static const int MAX\_SIZE = 100;

int arr[MAX\_SIZE];

int top;

public:

ArrayStack() {

top = -1;

}

bool isEmpty() {

return top == -1;

}

void push(int x) {

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

cout << "Stack Overflow\n";

return;

}

arr[++top] = x;

}

int pop() {

if (isEmpty()) {

cout << "Stack Underflow\n";

return -1;

}

return arr[top--];

}

void complement() {

for (int i = 0; i <= top; ++i) {

arr[i] = ~arr[i];

}

}

};

// Define a class for the stack implemented using a linked list

class LinkedListStack {

private:

Node\* top;

public:

LinkedListStack() {

top = nullptr;

}

bool isEmpty() {

return top == nullptr;

}

void push(int x) {

Node\* newNode = new Node;

newNode->data = x;

newNode->next = top;

top = newNode;

}

int pop() {

if (isEmpty()) {

cout << "Stack Underflow\n";

return -1;

}

int data = top->data;

Node\* temp = top;

top = top->next;

delete temp;

return data;

}

void complement() {

Node\* current = top;

while (current != nullptr) {

current->data = ~current->data;

current = current->next;

}

}

};

int main() {

// Using ArrayStack

ArrayStack arrayStack;

arrayStack.push(5);

arrayStack.push(10);

arrayStack.complement();

cout << "Array Stack elements after complement:\n";

while (!arrayStack.isEmpty()) {

cout << arrayStack.pop() << " ";

}

cout << endl;

// Using LinkedListStack

LinkedListStack linkedListStack;

linkedListStack.push(7);

linkedListStack.push(15);

linkedListStack.complement();

cout << "Linked List Stack elements after complement:\n";

while (!linkedListStack.isEmpty()) {

cout << linkedListStack.pop() << " ";

}

cout << endl;

return 0;

}

This program demonstrates how to create and manipulate stacks using both arrays and linked lists in C++. The complement() function complements each element in the stack.