1st Create a queue

#include <iostream>

#define MAX 100

Using namespace std;

class Queue {

private:

    int arr[MAX];

    int front;

    int rear;

    int size;

public:

    Queue() {

        front = 0;

        rear = -1;

        size = 0;

    }

    // Enqueue operation

    void enqueue(int value) {

        if (size == MAX) {

            cout << "Queue is full!\n";

            return;

        }

        rear = (rear + 1) % MAX;

        arr[rear] = value;

        size++;

        cout << value << " added to the queue.\n";

    }

    // Dequeue operation

    void dequeue() {

        if (isEmpty()) {

            cout << "Queue is empty!\n";

            return;

        }

        cout << arr[front] << " removed from the queue.\n";

        front = (front + 1) % MAX;

        size--;

    }

    // Peek the front element

    int peek() {

        if (isEmpty()) {

            cout << "Queue is empty!\n";

            return -1;

        }

        return arr[front];

    }

    // Check if the queue is empty

    bool isEmpty() {

        return size == 0;

    }

    // Display the queue

    void display() {

        if (isEmpty()) {

            cout << "Queue is empty!\n";

            return;

        }

        cout << "Queue elements: ";

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

            cout << arr[(front + i) % MAX] << " ";

        }

        cout << "\n";

    }

};

int main() {

    Queue q;

    int choice, value;

    do {

        cout << "\nQueue Operations:\n";

        cout << "1. Enqueue\n";

        cout << "2. Dequeue\n";

        cout << "3. Peek\n";

        cout << "4. Display\n";

        cout << "5. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

            cout << "Enter the value to enqueue: ";

            cin >> value;

            q.enqueue(value);

            break;

        case 2:

            q.dequeue();

            break;

        case 3:

            value = q.peek();

            if (value != -1) {

cout << "Front element: " << value << "\n";

            }

            break;

        case 4:

            q.display();

            break;

        case 5:

            cout << "Exiting...\n";

            break;

        default:

            cout << "Invalid choice!\n";

        }

    } while (choice != 5);

    return 0;

}

2nd Stack

#include <iostream>

#define MAX 1000 // Define the maximum size of the stack

class Stack {

private:

    int top;             // Index of the top element

    int arr[MAX];        // Array to store stack elements

public:

    Stack() { top = -1; } // Constructor initializes top to -1

    // Method to push an element onto the stack

    bool push(int value) {

        if (top >= (MAX - 1)) {

            std::cout << "Stack Overflow\n";

            return false;

        } else {

            arr[++top] = value;

            std::cout << value << " pushed onto the stack\n";

            return true;

        }

    }

    // Method to pop an element from the stack

    int pop() {

        if (top < 0) {

            std::cout << "Stack Underflow\n";

            return 0;

        } else {

            int value = arr[top--];

            return value;

        }

    }

    // Method to peek at the top element of the stack

    int peek() {

        if (top < 0) {

            std::cout << "Stack is Empty\n";

            return 0;

        } else {

            return arr[top];

        }

    }

    // Method to check if the stack is empty

    bool isEmpty() {

        return (top < 0);

    }

};

int main() {

    Stack stack;

    stack.push(10);

    stack.push(20);

    stack.push(30);

    std::cout << stack.pop() << " popped from the stack\n";

    std::cout << "Top element is: " << stack.peek() << "\n";

    std::cout << "Stack is " << (stack.isEmpty() ? "empty" : "not empty") << "\n";

    return 0;

}