#include<iostream>

#define MAX 10

using namespace std;

struct queue {

int data[MAX];

int front, rear;

};

class Queue {

struct queue q;

public:

Queue() {

q.front = q.rear = -1;

}

int isempty();

int isfull();

void enqueue(int);

int delqueue();

void display();

};

int Queue::isempty() {

return (q.front == -1) ? 1 : 0; // Check if front is -1 for empty queue

}

int Queue::isfull() {

return (q.rear == MAX - 1) ? 1 : 0; // Check if rear has reached max capacity

}

void Queue::enqueue(int x) {

if (q.rear == -1) {

q.front = 0; // Set front to 0 when first element is inserted

}

q.data[++q.rear] = x; // Increment rear and insert element

}

int Queue::delqueue() {

if (isempty()) {

cout << "Queue is empty!" << endl;

return -1; // Return a default value when queue is empty

}

return q.data[q.front++]; // Access and increment front

}

void Queue::display() {

if (isempty()) {

cout << "Queue is empty!" << endl;

return;

}

for (int i = q.front; i <= q.rear; i++) {

cout << q.data[i] << " "; // Print elements from front to rear

}

cout << endl;

}

int main() {

Queue obj;

int ch, x;

do {

cout << "\n1. Insert Job \n2. Delete Job \n3. Display \n4. Exit \nEnter your Choice: ";

cin >> ch;

switch (ch) {

case 1:

if (!obj.isfull()) {

cout << "\nEnter Data: ";

cin >> x;

obj.enqueue(x);

}

else

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

break;

case 2:

if (!obj.isempty()) {

cout << "\nDeleted Element = " << obj.delqueue() << endl;

}

else {

cout << "\nQueue is underflow!!!\n";

}

cout << "\nRemaining Jobs: ";

obj.display();

break;

case 3:

if (!obj.isempty()) {

cout << "\nQueue contains: ";

obj.display();

}

else

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

break;

case 4:

cout << "\nExiting Program....\n";

break;

}

} while (ch != 4);

return 0;

}