#include<iostream>

using namespace std;

class CircularQueue {

int size;

int \*arr;

int front;

int rear;

public:

CircularQueue() {

size = 100001;

arr = new int[size];

front = -1;

rear = -1;

}

bool isEmpty() {

if(front == -1) {

return true;

}

else {

return false;

}

}

bool isFull() {

if((front == 0 && rear == size-1) || rear == (front-1)%(size-1)) {

return true;

}

else {

return false;

}

}

void enqueue(int value) {

if(isFull()) {

cout<<"Queue is FULL"<<endl;

return;

}

if(front == -1) { // first element to push

front = rear = 0;

arr[rear] = value;

}

else if(rear== size-1 && front!=0) {

rear = 0;

arr[rear] = value;

}

else {

rear++;

arr[rear] = value;

}

}

void dequeue() {

if(isEmpty()) {

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

}

else if(front == rear) {

cout << "Dequeued element: " << arr[front] << endl;

front = -1;

rear = -1;

}

else if(front == size-1) {

cout << "Dequeued element: " << arr[front] << endl;

front = 0;

}

else {

cout << "Dequeued element: " << arr[front] << endl;

front++;

}

}

void peakQueue() {

if(isEmpty()) {

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

return;

}

cout << "Front element: " << arr[front] << endl;

}

void display() {

if(isEmpty()) {

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

return;

}

cout << "Queue elements: ";

int i = front;

do {

cout << arr[i] << " ";

i = (i + 1) % size;

} while(i != (rear + 1) % size);

cout << endl;

// Calculate size

int queueSize;

if(rear >= front) {

queueSize = rear - front + 1;

} else {

queueSize = size - front + rear + 1;

}

cout << "Size of Queue: " << queueSize << endl;

}

};

int main() {

CircularQueue q;

cout << "=== Circular Queue Operations ===" << endl;

q.enqueue(2);

q.enqueue(4);

q.enqueue(6);

q.enqueue(8);

q.display();

cout << "\nPeek operation:" << endl;

q.peakQueue();

cout << "\nDequeue operations:" << endl;

q.dequeue();

q.display();

q.dequeue();

q.display();

q.enqueue(10);

q.enqueue(12);

q.enqueue(14);

q.display();

return 0;

}

