# Practical 3 Source Code:-

#include <iostream>

using namespace std;

#define MAX\_SIZE 100 class CircularQueue { public: CircularQueue() {

front = -1; rear = -1;

}

bool isFull() {

return (rear + 1) % MAX\_SIZE == front;

}

bool isEmpty() { return front == -1;

}

void enqueue(int data) {

if (isFull()) {

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

return;

}

if (isEmpty()) { front = rear = 0;

} else {

rear = (rear + 1) % MAX\_SIZE;

}

queue[rear] = data; cout << data << " enqueued.\n";

}

int dequeue() { if (isEmpty()) { cout << "Queue is empty!" << endl;

return -1;

}

int data = queue[front]; if (front == rear) { front = rear = -1;

} else {

front = (front + 1) % MAX\_SIZE;

}

return data;

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

for (int i = front; i != rear; i = (i + 1) % MAX\_SIZE) { cout << queue[i] << " ";

}

cout << queue[rear] << endl;

}

private: int queue[MAX\_SIZE]; int front, rear;

};

int main() { CircularQueue cq;

int choice, data;

while (true) {

cout << "\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) { case 1:

cout << "Enter data to enqueue: ";

cin >> data; cq.enqueue(data); break; case 2:

data = cq.dequeue(); if (data != -1) {

cout << data << " dequeued.\n";

}

break; case 3: cq.display(); break; case 4: exit(0); default:

cout << "Invalid choice!" << endl;

}

}

return 0;

}

# Output:-

