**ASSIGNMENT-5**

**Implement circular queue.**

**CODE:**

public class CQueue {

int SIZE = 5; // Size of Circular Queue

int front, rear;

int items[] = new int[SIZE];

CQueue() {

front = -1;

rear = -1;

}

boolean isFull() {

if (front == 0 && rear == SIZE - 1) {

return true;

}

if (front == rear + 1) {

return true;

}

return false;

}

boolean isEmpty() {

if (front == -1)

return true;

else

return false;

}

void enQueue(int element) {

if (isFull()) {

System.out.println("Queue is full");

} else {

if (front == -1)

front = 0;

rear = (rear + 1) % SIZE;

items[rear] = element;

System.out.println("Inserted " + element);

}

}

int deQueue() {

int element;

if (isEmpty()) {

System.out.println("Queue is empty");

return (-1);

} else {

element = items[front];

if (front == rear) {

front = -1;

rear = -1;

}

else {

front = (front + 1) % SIZE;

}

return (element);

}

}

void display() {

int i;

if (isEmpty()) {

System.out.println("Empty Queue");

} else {

System.out.println("Front -> " + front);

System.out.println("Items -> ");

for (i = front; i != rear; i = (i + 1) % SIZE)

System.out.print(items[i] + " ");

System.out.println(items[i]);

System.out.println("Rear -> " + rear);

}

}

public static void main(String[] args) {

CQueue q = new CQueue();

q.deQueue();

q.enQueue(1);

q.enQueue(2);

q.enQueue(3);

q.enQueue(4);

q.enQueue(5);

q.enQueue(6);

q.display();

int elem = q.deQueue();

if (elem != -1) {

System.out.println("Deleted Element is " + elem);

}

q.display();

q.enQueue(7);

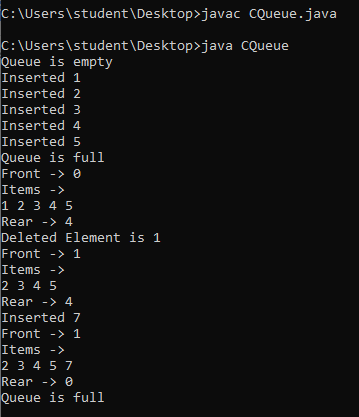
q.display();

q.enQueue(8);

}

}

**OUTPUT:**

****