**Practical No.3**

**Circular Queue**

**Problem Statement:-**

Implement Circular Queue using Array. Perform following operations on it. a) Insertion (Enqueue) b) Deletion (Dequeue) c) Display (Note: Handle queue full condition by considering a fixed size of a queue.)

**Program Code:-**

#include <iostream>

using namespace std;

int cqueue[5];

int front = -1, rear = -1, n=5;

**void insertCQ(int val) {**

if ((front == 0 && rear == n-1) || (front == rear+1))

{

cout<<"Queue Overflow \n";

return;

}

if (front == -1)

{

front = 0;

rear = 0;

} else {

if (rear == n - 1)

rear = 0;

else

rear = rear + 1;

}

cqueue[rear] = val ;

}

**void deleteCQ() {**

if (front == -1) {

cout<<"Queue Underflow\n";

return ;

}

cout<<"Element deleted from queue is : "<<cqueue[front]<<endl;

if (front == rear) {

front = -1;

rear = -1;

} else {

if (front == n - 1)

front = 0;

else

front = front + 1;

}

}

**void displayCQ()** {

int f = front, r = rear;

if (front == -1) {

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

return;

}

cout<<"Queue elements are :\n";

if (f <= r) {

while (f <= r){

cout<<cqueue[f]<<" 🡪";

f++;

}

} else {

while (f <= n - 1) {

cout<<cqueue[f]<<" 🡪";

f++;

}

f = 0;

while (f <= r) {

cout<<cqueue[f]<<" 🡪";

f++;

}

}

cout<<endl;

}

**int main()**

{

int ch, val;

cout<<"1)Insert\n";

cout<<"2)Delete\n";

cout<<"3)Display\n";

cout<<"4)Exit\n";

do {

cout<<"Enter choice : "<<endl;

cin>>ch;

switch(ch) {

case 1:

cout<<"Input for insertion: "<<endl;

cin>>val;

insertCQ(val);

break;

case 2:

deleteCQ();

break;

case 3:

displayCQ();

break;

case 4:

cout<<"Exit\n";

break;

default: cout<<"Incorrect!\n";

}

} while(ch != 4);

return 0;

}