**AIM:** Pizza parlor accepting maximum 10 order .Orderare served on first come first serve basis .order once placed can’t be cancel .write c++ to simulate the system using circular Queue

**Objective:** Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**

**Theory :** Operations on Circular Queue:

* **Front:** Get the front item from queue.
* **Rear:** Get the last item from queue.
* **enQueue(value)**This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.
* **deQueue()** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position.

**Source Code:**

**#include<iostream>**

**#include<queue>**

**using namespace std;**

**int qfull(int \*m,int \*n) //Check if queue is full**

**{**

**int size=10;**

**if(\*m==((\*n+1)%size))**

**{**

**return 1;**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**int qempty(int \*m,int \*n) //Check if queue is empty**

**{**

**if(\*m==-1)**

**return 1;**

**else**

**return 0;**

**}**

**void enqueue(int a[],int \*m,int \*n)**

**{ int size=10;**

**if(qfull(m,n))**

**{**

**cout<<"Orders are Pending"<<endl;**

**}**

**else**

**{**

**if(\*m==-1)**

**{**

**\*m=0;**

**\*n=0;**

**cin>>a[\*n];**

**}**

**else**

**{**

**\*n=(\*n+1)%size;;**

**cin>>a[\*n];**

**}**

**}**

**}**

**void dequeue(int a[],int \*m,int \*n)**

**{ int size=10;**

**int order;**

**if(qempty(m,n))**

**{**

**cout<<"No orders are remain to serve"<<endl;**

**}**

**else**

**{**

**order=a[\*m];**

**if(\*m==\*n)**

**{**

**\*m=-1;**

**\*n=-1;**

**}**

**else**

**{**

**\*m=(\*m+1)%size;**

**}**

**cout<<order<<"\tOrder Served"<<endl;**

**}**

**}**

**int main()**

**{**

**int front=-1;**

**int rear=-1;**

**int \*m=&front;**

**int \*n=&rear;**

**int a[10],k,op;**

**while(1){**

**cout<<"Operations"<<endl;**

**cout<<"1.Accept Order \t 2.Serve Order \t 3.Exit "<<endl;**

**cin>>k;**

**switch(k)**

**{**

**case 1:**

**enqueue(a,m,n);**

**break;**

**case 2:**

**dequeue(a,m,n);**

**break;**

**case 3:**

**exit(0);**

**default :**

**cout<<"Incorrect wrong operation"<<endl;**

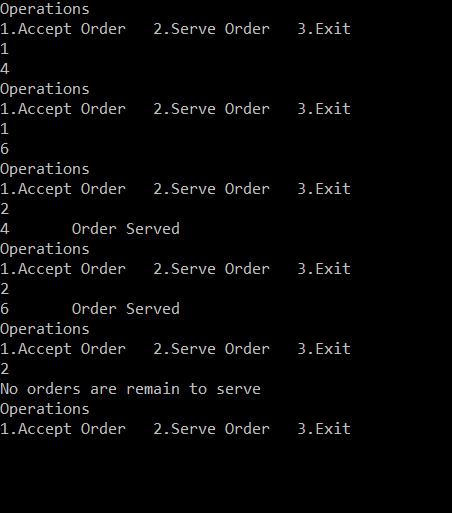
**break;**

**}**

**}**

**return 0;**

**}**

****