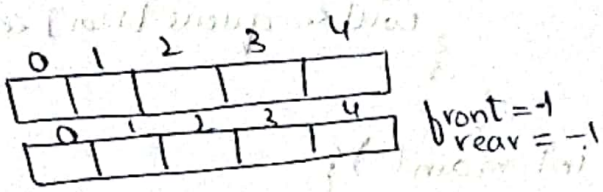
Assignment # 3

Submitted by: Tayba Asghar
Registration No.: SP22-BCS-077
Submitted to: Mam Yasmeen Jana
Section: Section - B
Topic: Queue

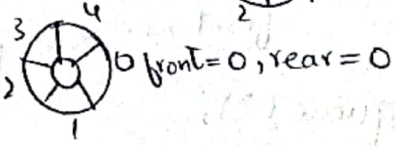
Queue

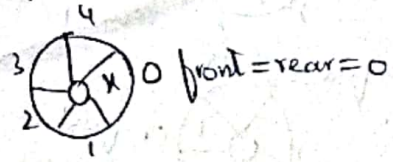
Circular Queue

```
#include <iostream>
#include <stack>
using namespace std;
int queue[5], N=5
int front=-1, rear=-1;
```

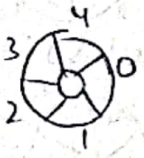


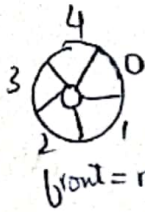

```
void enqueue (int x){
    if ((rear+1)%N == front){
        cout << "Queue is full" << endl;
    }
    else{
        if (front == -1 && rear == -1){
            front = rear = 0;
        }
        else{
            rear = (rear+1)%N;
        }
        queue[rear] = x;
    }
```





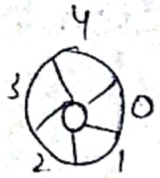

```
void dequeue(){
    if (front == -1 && rear == -1){
        cout << "Queue is empty" << endl;
    }
    else if (front == rear){
        cout << "Dequeued element =" << queue[front] << endl;
        front = rear = -1;
    }
    else{
        cout << "Dequeued element=" << queue[front] << endl;
```



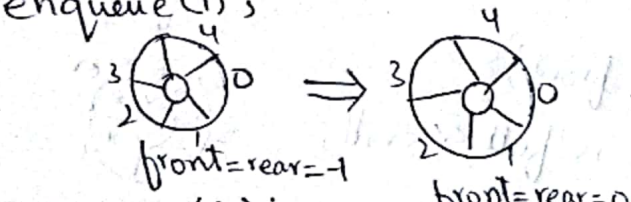


Scanned with CamScanner

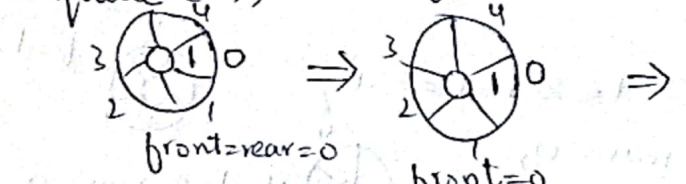
```
        front = (front+1) % N;
    }
}
void display(){
    int i = front;
    if (front == -1 && rear == -1){
        cout<<"Queue is empty "<<endl;
    }else{
        cout<< "Queue elements:";
        while(i != rear){
            cout<<queue[i]<<" ";
            i = (i+1)%N;
        }
        cout<<queue[rear]<<endl;
    }
}
int main(){
    enqueue(1);
    enqueue(2);
    enqueue(3);
    enqueue(4);
    enqueue(5);
    display();
```



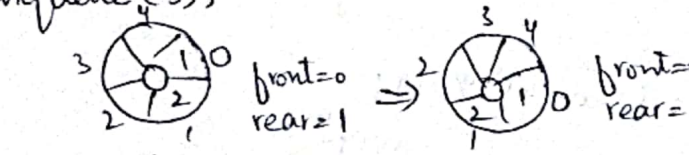
front = rear = -1



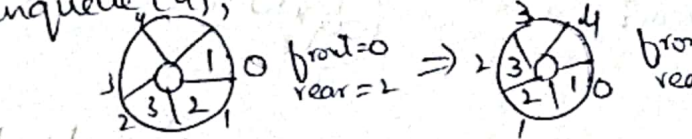
front=rear=-1 ⇒ front=rear=0 ⇒ front=0, rear=0, rear[0]=1



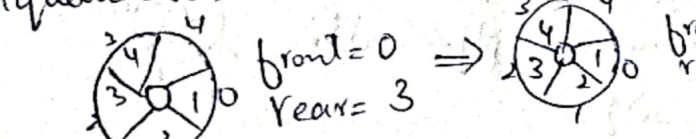
front=rear=0 ⇒ front=0, rear=1, rear[1]=2



front=0, rear=1 ⇒ front=0, rear=2 ⇒ front=0, rear=2, rear[2]=3



front=0, rear=2 ⇒ front=0, rear=3 ⇒ front=0, rear=3, rear[3]=4

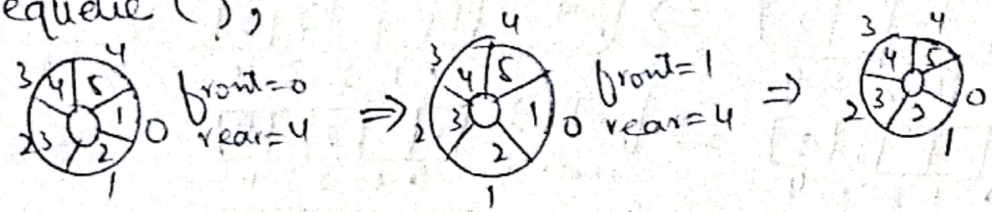


front=0, rear=3 ⇒ front=0, rear=4 ⇒ front=0, rear=4, rear[4]=5

Queue elements: 1 2 3 4 5

Scanned with CamScanner

dequeue();



Dequeued element = 1

display();

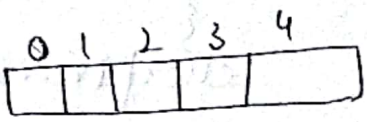
Queue elements : 2 3 4 5

return 0;

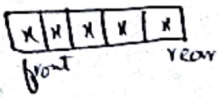
}

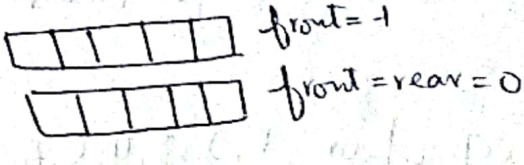
Deque

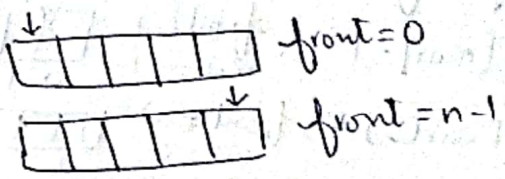
```
#include <iostream>
using namespace std;
int deque[5], n=5, front=rear=-1;
```



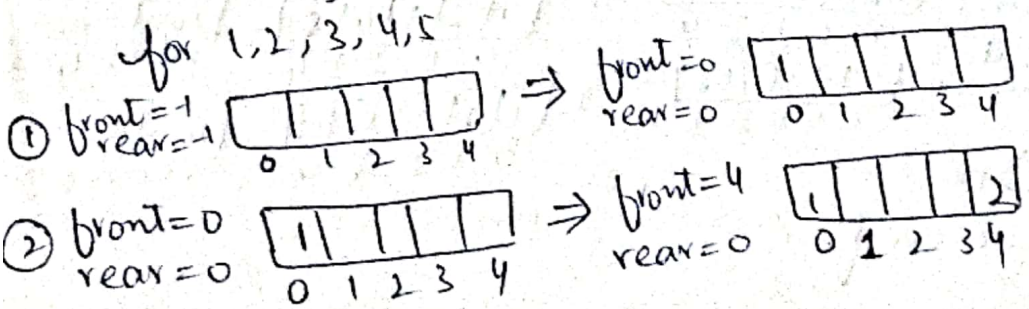

```
void insertFront(int val){
    if((front==0 && rear==n-1) || (front==rear+1))
        cout << "Deque Overflow" << endl;
    else{
        if(front==-1){
            front=0;
            rear=0;
        }
        else if(front==0)
            front=n-1;
        else
            front--;
        deque[front]=val;
    }
```





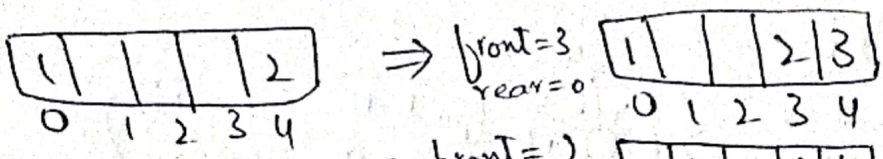


for 1, 2, 3, 4, 5

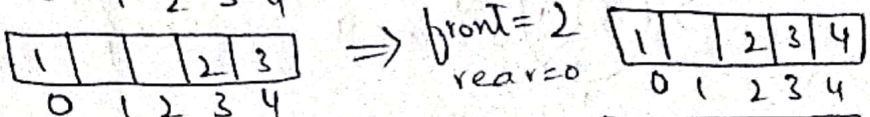


Scanned with CamScanner

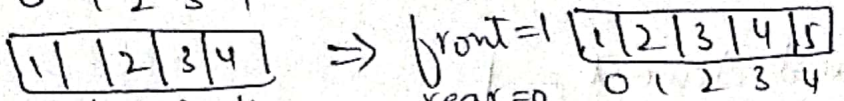
③ front=4, rear=0 ⇒ front=3, rear=0



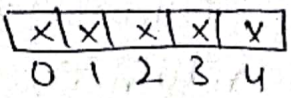
④ front=3, rear=0 ⇒ front=2, rear=0

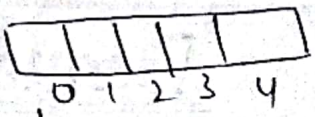


⑤ front=2, rear=0 ⇒ front=1, rear=0

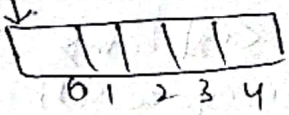



```
Void insertRear(int val){
    if(front==0 && rear=n-1) || (front ==rear+1))
        cout<< "Deque Overflow"<<endl;
    else{
        if(rear == -1){
            front=0;
            rear=0;
        }
        else if(rear==n-1)
            rear=0;
        else
            rear++;
        deque[rear]=val;
    }
}
```

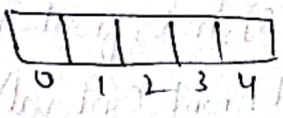




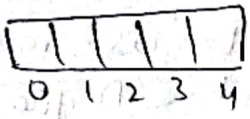
front = rear = -1



front = rear = 0



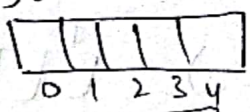
rear = 4



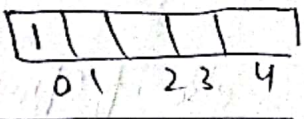
rear = 0

eg. for 1, 2, 3, 4, 5, 6

deque[rear]=1 ⇒ front=-1, rear=-1

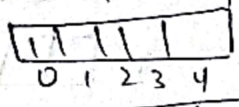


⇒

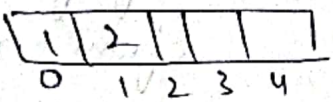


front = rear = 0

insertRear(1) ⇒ front=0, rear=0

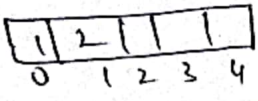


⇒

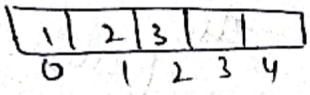


front=0, rear=1

insertRear(3) ⇒ front=0, rear=1

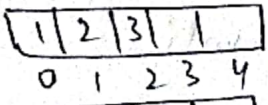


⇒

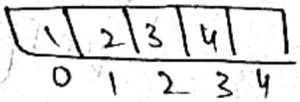


front=0, rear=2

insertRear(4) ⇒ front=0, rear=2

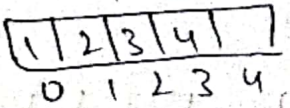


⇒

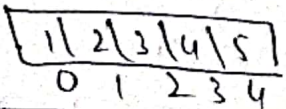


front=0, rear=3

insertRear(5) ⇒ front=0, rear=3

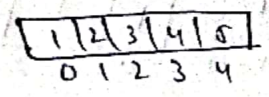


⇒

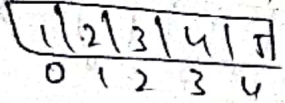


front=0, rear=4

insertRear(6) ⇒ front=0, rear=4



⇒



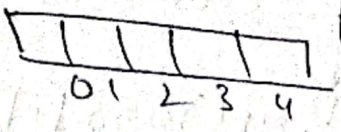
front=0, rear=5

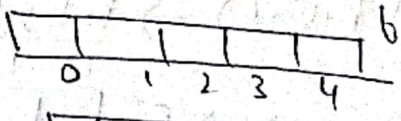
rear > n

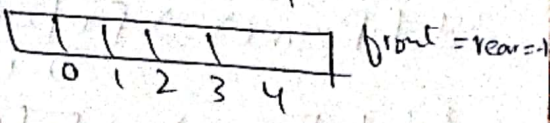
"Deque overflow".

Scanned with CamScanner

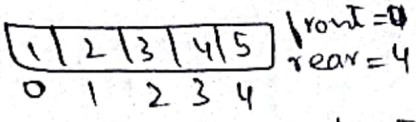
```
void deleteFront(){
    if(front==-1){
        cout<<"Deque Underflow"<<endl;
    }
    else{
        cout<<"Deleted element is"<<deque[front]<<endl;
        if(front==rear){
            front=rear=-1;
        }
        else if(front==n-1)
            front=0;
        else
            front++;
    }
}
```

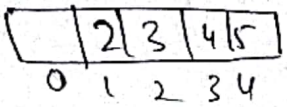




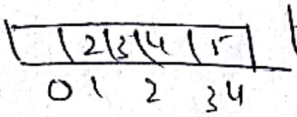


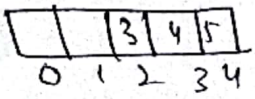
e.g. 1,2,3,4,5

deleteFront() ⇒ 

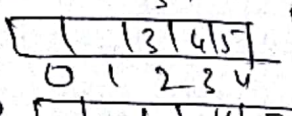
⇒ 

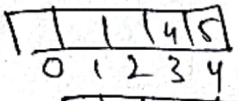
front=1
rear=4

deleteFront() ⇒ 

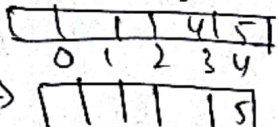
⇒ 

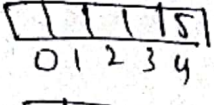
front=2
rear=4

deleteFront() ⇒ 

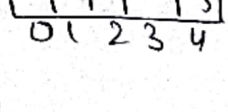
⇒ 

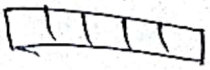
front=3
rear=4

deleteFront() ⇒ 

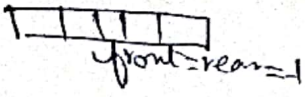
⇒ 

front=4
rear=4

deleteFront() ⇒ 

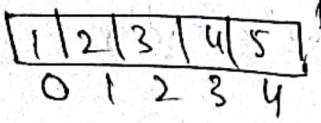
⇒ 
front=-1
rear=-1

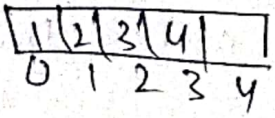
```
void deleteRear(){
    if(rear==-1)
        cout<<"Deque Underflow"<<endl;
    else{
        cout<<"The delete element is"<<deque[rear]<<endl;
        if(front==rear)
            front=rear=-1;
        else if(rear==0)
            rear=n-1;
        else
            rear--;
    }
}
```



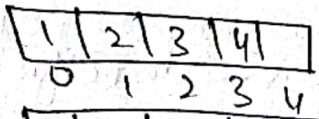
Scanned with CamScanner

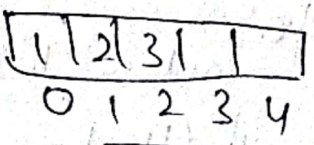
e.g. elements = 1, 2, 3, 4, 5

deleteRear() ⇒ 

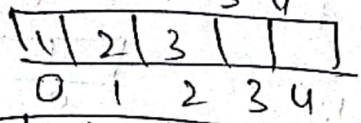
front=0 rear=4 ⇒ 

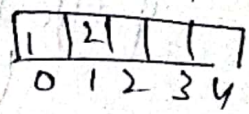
front=0 rear=3

deleteRear() ⇒ 

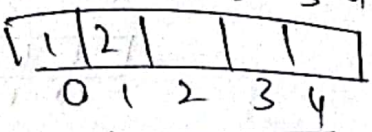
front=0 rear=3 ⇒ 

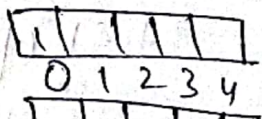
front=0 rear=2

deleteRear() ⇒ 

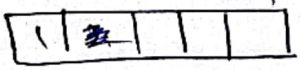
front=0 rear=2 ⇒ 

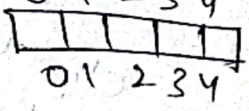
front=0 rear=1

deleteRear() ⇒ 

front=0 rear=1 ⇒ 

front=0 rear=0

deleteRear() ⇒ 

front=0 rear=0 ⇒ 
front=-1 rear=-1

deleteRear() ⇒ Deque Underflow

Scanned with CamScanner