Day 18

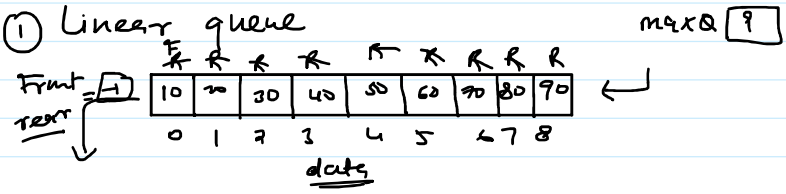
07 April 2025 08:30 AM

Queue :

Types of queue :-

1. Linear queue (Queue)
2. Circular queue
3. multi queue
4. Double ended queue
5. priority queue.

1) Linear queue

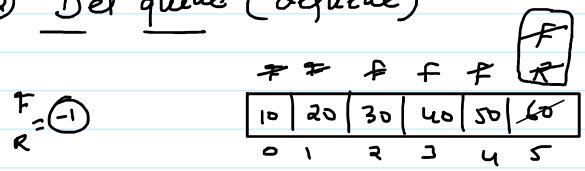


1. Add
2. del

1) Add queue :-

```
num 100
if(rear == maxQ-1)
    overflow, Return
if( front == -1)
    Front = rear = 0;
else
    rear++
data[rear] = num;
```

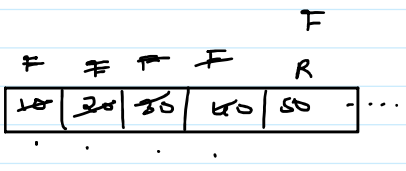
2) Del queue (dequeue)



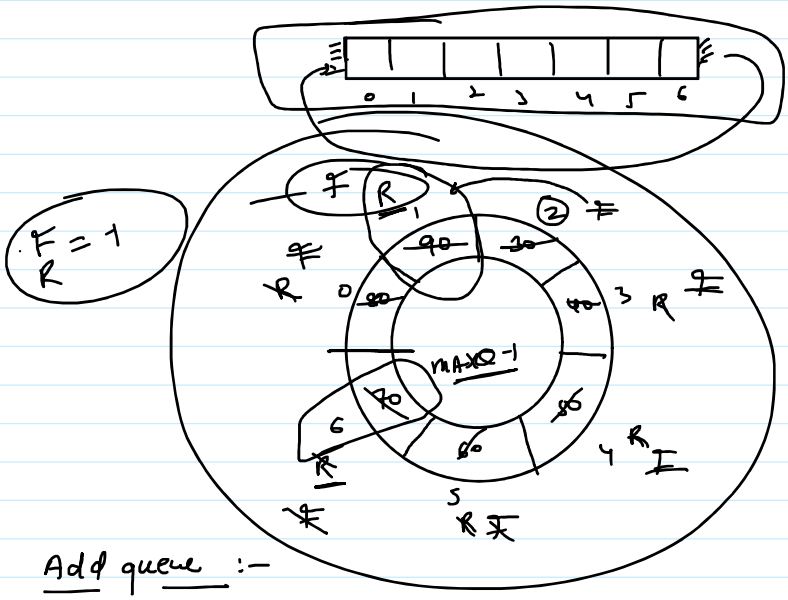

```
if(front == -1)
    Underflow, return.
if(front == rear)
    front = rear = -1
else
    front ++
```

```
#include<iostream>
#define MAXQ 10
using namespace std;
class Queue{
```

```
    int data[MAXQ];
    int rear,front;
    public:
        Queue()
        {
            rear = front = -1;
        }
        void addQueue(int num)
        {
            if(rear == MAXQ-1)
            {
                cout<<"Overflow\n";
                return;
            }
            if(front == -1)
                front = rear = 0;
            else
                rear++;
            data[rear] = num;
        }
        void delQueue()
        {
            if(rear==-1)
            {
                cout<<"Underflow\n";
                return;
            }
            if(rear == front)
                rear = front = -1;
            else
                front++;
        }
        int first(){
            return data[front];
        }
        int last()
        {
            return data[rear];
        }
        bool isEmpty()
        {
            return front == -1 ? true : false;
        }
};
int main()
{
    Queue q1;
    q1.addQueue(10);
    q1.addQueue(20);
    q1.addQueue(30);
    q1.addQueue(40);
    q1.addQueue(50);
    q1.addQueue(60);
    q1.addQueue(70);
    q1.addQueue(80);
    q1.addQueue(90);
    q1.addQueue(100);
    q1.addQueue(110);
    cout<<q1.first()<<" "<<q1.last()<<endl;
    q1.delQueue();
    cout<<q1.first()<<" "<<q1.last()<<endl;
    while(! q1.isEmpty())
    {
        cout<<q1.first()<<" ";
        q1.delQueue();
    }
}
```



② Circular queue :-



Add queue :-

0 6

```
if ( front == 0 && rear == maxQ-1 || rear == front-1 )
        overflow, Return

if ( front == -1)
        front = rear = 0;
else if ( rear == maxQ-1)
        rear = 0;
else
        rear++;
data[rear] = num;
```

Del Queue

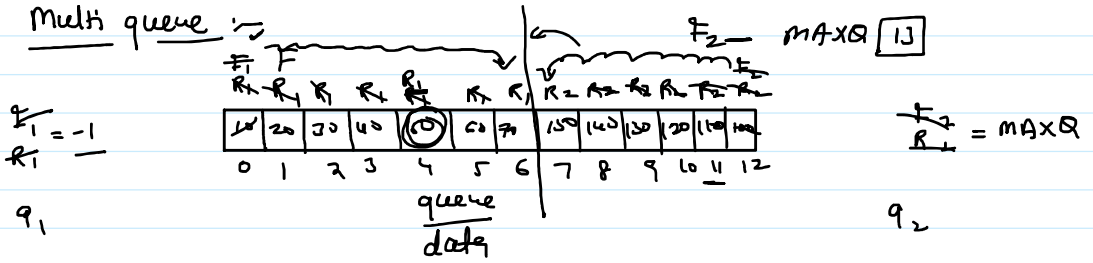
```
if (front == -1)
        Under flow, Return

if ( front == rear)
        front = rear = -1;
else if (front == maxQ-1)
        front = 0;
else
        front++;
```

```
#include<iostream>
#define MAXQ 10
using namespace std;
class CircularQueue{
```

```
    int data[MAXQ];
    int rear,front;
    public:
        CircularQueue()
        {
            rear = front = -1;
        }
        void addQueue(int num)
        {
            if(rear == MAXQ-1&&front==0 || rear==front-1)
            {
                cout<<"Overflow\n";
                return;
            }
            if(front == -1)
                front = rear = 0;
            else if(rear==MAXQ-1)
                rear = 0;
            else
                rear++;
            data[rear] = num;
        }
        void delQueue()
        {
            if(rear==-1)
            {
                cout<<"Underflow\n";
                return;
            }
            if(rear == front)
                rear = front = -1;
            else if(front == MAXQ-1)
                front = 0;
            else
                front++;
        }
        int first(){
            return data[front];
        }
        int last()
        {
            return data[rear];
        }
        bool isEmpty()
        {
            return front == -1 ? true : false;
        }
};
int main()
{
    CircularQueue q1;
    q1.addQueue(10);
    q1.addQueue(20);
    q1.addQueue(30);
    q1.addQueue(40);
    q1.addQueue(50);
    q1.addQueue(60);
    q1.addQueue(70);
    q1.addQueue(80);
    q1.addQueue(90);
    q1.addQueue(100);
    q1.addQueue(110);
    q1.delQueue();
    q1.addQueue(110);
    // cout<<q1.first()<<" "<<q1.last()<<endl;
    // q1.delQueue();
    // cout<<q1.first()<<" "<<q1.last()<<endl;
    while(! q1.isEmpty())
    {
        cout<<q1.first()<<" ";
        q1.delQueue();
    }
}
```

Multi queue :-



$f_1 = -1$
r_1

f_2
$r_2 = MAXQ$

q_1 q_2

✓ Add q_1

```
if (Rear1 == Rear2 + 1)
    overflow, Return
if( Rear1 == -1)
    rear1 = front1 = 0
else
    rear1++
data[rear1] = num ;
```

Add q_2

```
if( Rear1 == Rear2 - 1 )
    overflow, Return.
if( Rear2 == MAXQ)
    rear2 = front2 = MAXQ-1
else
    rear2--;
data[rear2] = num;
```

✓ Del queue 1

```
if( Front1 == -1)
    underflow return
if( front1 == Rear1)
    front1 = Rear1 = -1;
else
    front1++ ;
```

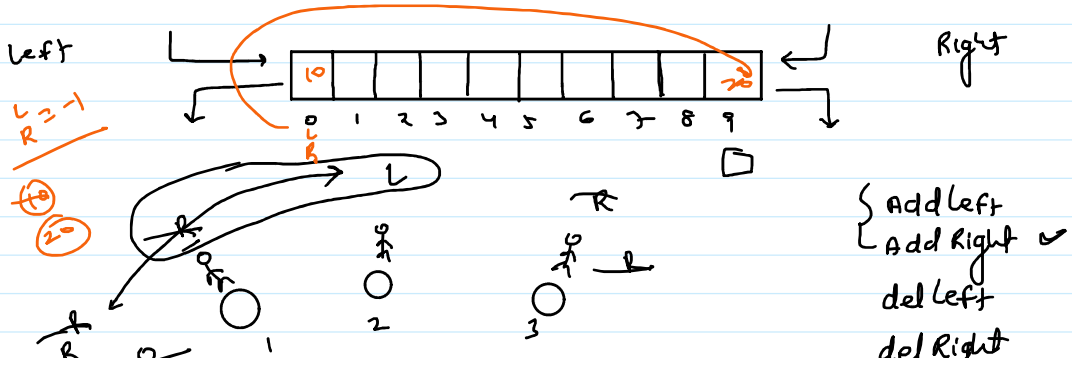
Del queue 2

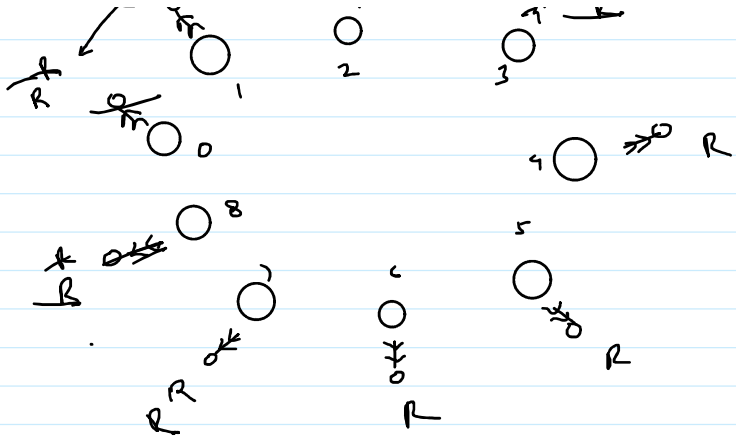
```
if( Front2 == MAXQ )
    underflow, Return
if( front2 == Rear2)
    front2 = Rear2 = MAXQ
else
    front2-- ;
```

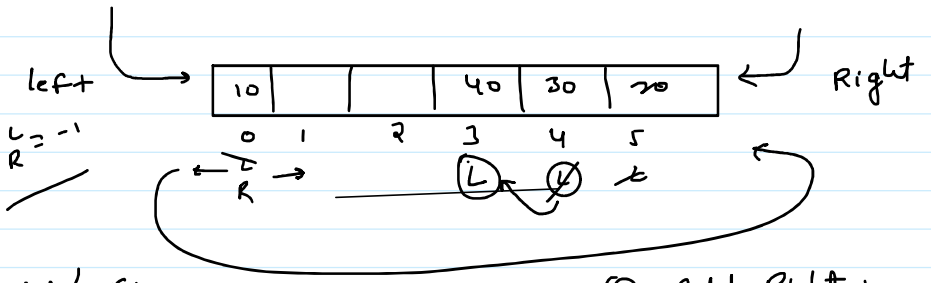
```
#include<iostream>
#define MAXQ 10
using namespace std;
class MultiQueue{
    int data[MAXQ];
    int rear1,front1,front2,rear2;
    public:
        MultiQueue()
        {
            rear1 = front1 = -1;
            rear2 = front2 = MAXQ;
        }
        void addQueue1(int num)
        {
            if(rear1 == rear2-1)
            {
                cout<<"Overflow\n";
                return;
            }
            if(front1 == -1)
                front1 = rear1 = 0;
            else
```

```
            rear1++;
        data[rear1] = num;
    }
    void delQueue1()
    {
        if(rear1==-1)
        {
            cout<<"Underflow\n";
            return;
        }
        if(rear1 == front1)
            rear1 = front1 = -1;
        else
            front1++;
    }
    void addQueue2(int num)
    {
        if(rear1 == rear2-1)
        {
            cout<<"Overflow\n";
            return;
        }
        if(front2 == MAXQ)
            front2 = rear2 = MAXQ-1;
        else
            rear2--;
        data[rear2] = num;
    }
    void delQueue2()
    {
        if(rear2==MAXQ)
        {
            cout<<"Underflow\n";
            return;
        }
        if(rear2 == front2)
            rear2 = front2 = MAXQ;
        else
            front2--;
    }
};
int main()
{
    MultiQueue q1;
    q1.addQueue1(10);
    q1.addQueue1(20);
    q1.addQueue1(30);
    q1.addQueue1(40);
    q1.addQueue1(50);
    q1.addQueue2(100);
    q1.addQueue2(200);
    q1.addQueue2(300);
    q1.addQueue2(400);
    q1.addQueue2(500);
    q1.addQueue1(150);
}
```

Double ended queue :-







Add Left

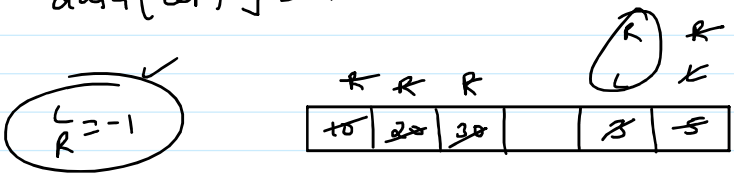
Same

① Add Right :- 10

```
if( left == 0 && Right == MAXQ-1 )
        Right == left - 1
```

```
if (left = -1)
        left = Right = MAXQ-1;
else if (Right == MAXQ-1)
        Right = 0;
else
        Right ++;
data [Right] = num;
```

```
if (left == -1)
        left = Right = 0
else if( left == 0)
        left = MAXQ-1;
else
        left --
data [left] = num
```



① Del Left

```
if (left == -1)
        underflow, Return
if (left == Right)
        left = Right = -1;
else if (left == MAXQ-1)
        left = 0
else
        left ++
```

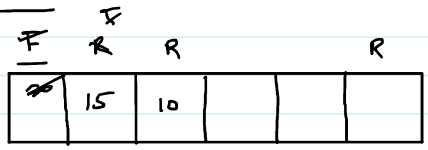
Del Right

```
if( left == -1)
        underflow, Return
if( left == Right)
        left = Right = -1;
else if ( Right == 0)
        Right = MAXQ-1
else
        Right --;
```

```
#include<iostream>
#define MAXQ 10
using namespace std;
class DoubleEndedQueue{
    int data[MAXQ];
    int right,left;
    public:
        DoubleEndedQueue()
        {
            right = left = -1;
        }
        void addRight(int num)
        {
            if(right == MAXQ-1&&left==0 || right==left-1)
            {
                cout<<"Overflow\n";
                return;
            }
            if(left == -1)
                left = right = 0;
            else if(right==MAXQ-1)
                right = 0;
            else
                right++;
            data[right] = num;
        }
        void addLeft(int num)
        {
            if(right == MAXQ-1&&left==0 || right==left-1)
            {
                cout<<"Overflow\n";
                return;
            }
            if(left == -1)
                left=right=0;
            else if(left == 0)
                left = MAXQ-1;
            else
                left--;
            data[left]=num;
        }
        void delLeft()
        {
            if(right==-1)
            {
                cout<<"Underflow\n";
                return;
            }
            if(right == left)
                right = left = -1;
            else if(left == MAXQ-1)
                left = 0;
            else
                left++;
        }
        void delRight()
        {
            if(right==-1)
            {
                cout<<"Underflow\n";
                return;
            }
            if(right == left)
                right = left = -1;
            else if(right == 0)
                right = MAXQ-1;
            else
                right--;
        }
        int left_element(){
            return data[left];
        }
        int right_element()
```

```
        {
            return data[right];
        }
        bool isEmpty()
        {
            return left == -1 ? true : false;
        }
        void output()
        {
            for(int i=left; i!=right ; i++)
            {
                cout<<data[i]<<" ";
                if(i==MAXQ-1)
                    i=-1;
            }
            cout<<data[right]<<endl;
        }
};
int main()
{
    DoubleEndedQueue q1;
    q1.addLeft(10);
    q1.addLeft(20);
    q1.addLeft(30);
    q1.addRight(100);
    q1.addRight(200);
    q1.addRight(300);
    q1.output();
}
```

Priority queue :→



MAXQ 6

f = -1
R = -1

i
1
0

j
queue

10
5
100
20

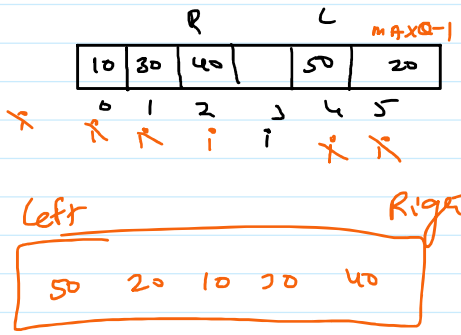
n 20 15

```
if (Rear == MAXQ-1)
    Overflow, Return;
if (front == -1)
{
    front = Rear = 0
    data[Rear] = num;
    return;
}
Rear++
for(j = Rear-1; j>=front && data[j] < num; j--)
{
    data[j+1] = data[j];
}
data[j+1] = num;
```

```
#include<iostream>
#define MAXQ 10
using namespace std;
class PriorityQueue{
    int data[MAXQ];
    int front,rear;
    public:
        PriorityQueue()
        {
            front = rear = -1;
        }
        void addQueue(int num)
        {
            if(rear == MAXQ-1)
```

```
            {
                cout<<"Overflow\n";
                return;
            }
            if(front==-1)
            {
                front=rear=0;
                data[rear]=num;
                return;
            }
            rear++;
            int j;
            for(j=rear-1 ; j>=front && data[j]<num ;j--)
            {
                data[j+1] = data[j];
            }
            data[j+1] = num;
        }
        void delQueue()
        {
            if(front == -1)
            {
                cout<<"Underflow\n";
                return;
            }
            if(front == rear)
                front = rear = -1;
            else
                front++;
        }
        bool isEmpty()
        {
            return front == -1 ? true : false;
        }
        void output()
        {
            for(int i=front ; i<=rear ; i++)
            {
                cout<<data[i]<<" ";
            }
            cout<<endl;
        }
};
int main()
{
    PriorityQueue q1;
    q1.addQueue(10);
    q1.addQueue(5);
    q1.addQueue(20);
    q1.addQueue(15);
    q1.output();
    q1.delQueue();
    q1.output();

}
```



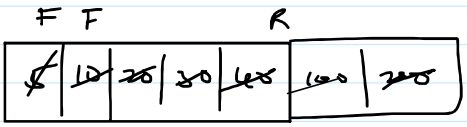
ql (10)
ql (20)
qr (30)
qr (40)
ql (50)

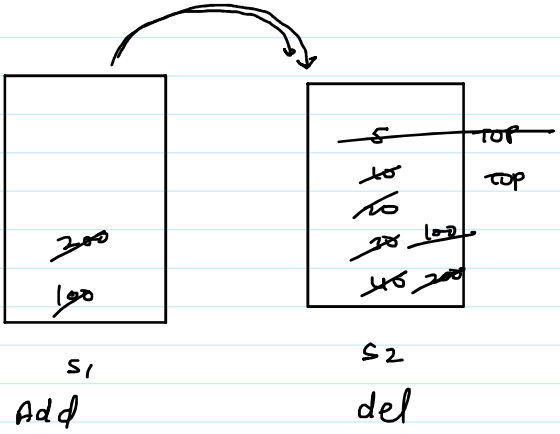
```
#include<iostream>
#include<queue>
```

```
using namespace std;
int main()
{
    queue<int> q1;
    q1.push(10);
    q1.push(20);
    q1.push(30);
    q1.push(40);
    cout<<q1.front()<<" "<<q1.back()<<endl;
    q1.pop();
    cout<<q1.front()<<" "<<q1.back()<<endl;
}
```

```
#include<iostream>
#include<queue>
using namespace std;
int main()
{
    priority_queue<int> q1;
    q1.push(10);
    q1.push(5);
    q1.push(20);
    q1.push(15);
    cout<<q1.top()<<endl;
    q1.pop();
    cout<<q1.top()<<endl;
}
```

```
#include<iostream>
#include<deque>
using namespace std;
int main()
{
    deque<int> q1;
    q1.push_back(20);
    q1.push_front(10);
    q1.pop_back();
    q1.pop_front();
}
```





H.W

leetcode - 232

leetcode - 225