Lab Assignment 03

Name: Anam Batool

Roll no: SP22-BCS-112

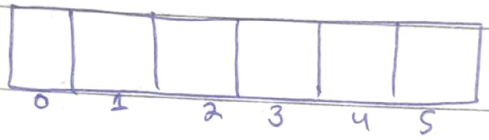
Section: B

Subject: DSA Lab

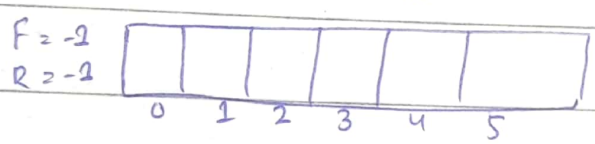
Queue

Linear Queue:

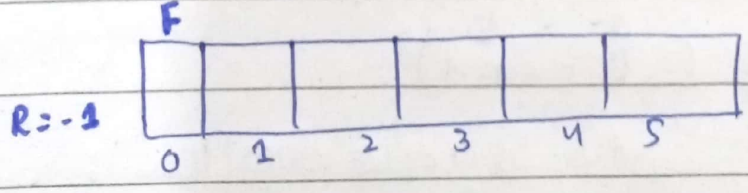
int queue[6], n=6;



int F=-1, R=-1;

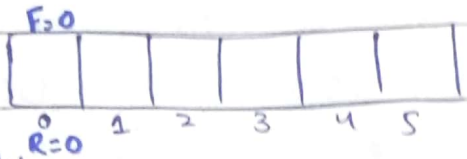



```
void insert()
{
    int value;
    if(R==n-1)        // R=-1, n=5 → condition false go to else part
    {
        cout<<" Queue overflow";
    }
else
    if (F==-1)
        f=0;
```

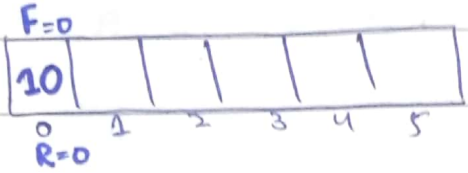


cin >> value; (10) Take input from user

rear ++;



queue[rear] = value; (rear = 0, value = 10)



Now go back to if condition, if it is false then move to else condition

if (R = n-1) (R = 0, n-1 = 5)

{

cout << "Queue overflow";

}

else

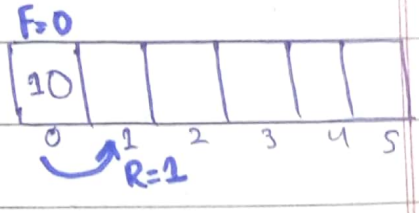
if (f == -1) This is false as front = 0

f = 0;

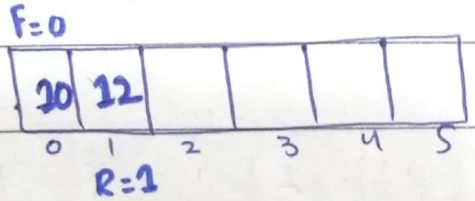
else

cin >> value; (12)

rear ++;



queue[rear] = value; (rear = 1, value = 12)



Now again back to else if condition

if (R == n-1) (R = 1, n-1 = 5)

{

cout << "Queue overflow";

}

else

if (front == -1) → condition false

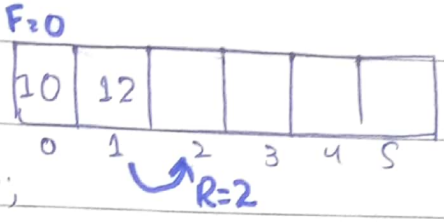
front = 0;

else

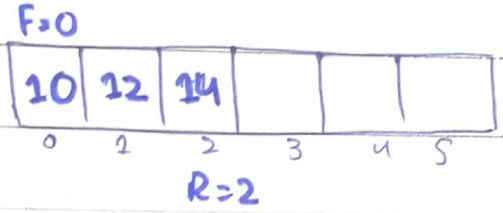
14

cin >> value;

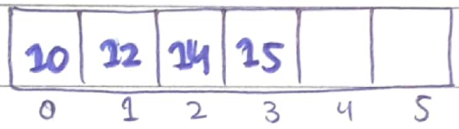
rear ++;



queue[rear] = value;



Repeat these steps untill end;



else

if (front == -1) → condition false

front = 0;

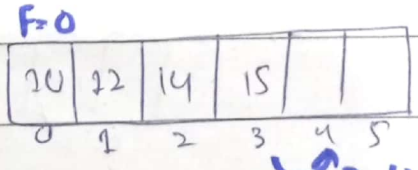
else

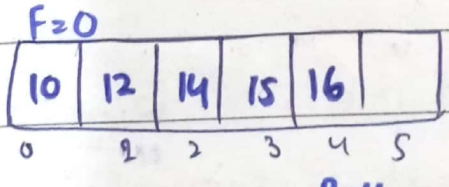
16

cin >> value;

rear ++;

queue[rear] = value;





Again;

else if (front == -1) → condition false

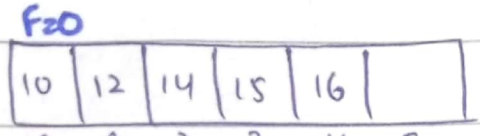
front = 0;

else

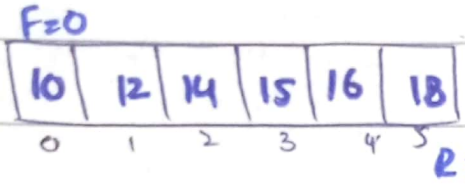
18

cin >> value;

rear ++;



queue[rear] = value;



Go back to if condition

if (R == n-1) → True so else condition will not run

{

cout << "Queue overflow";

}

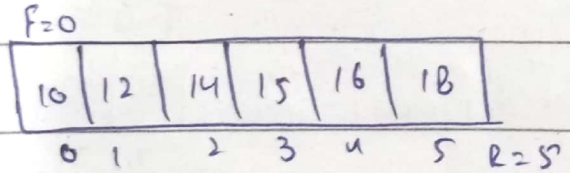
"Queue overflow" → print on screen.

void display()

{

if (F == -1) → condition false

{ cout << "Queue is empty";

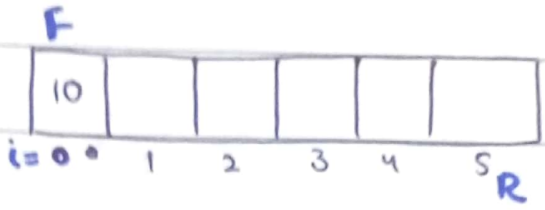


else

for (int i = F ; i <= R ; i ++)

cout << queue[i]; 10

cout << endl;



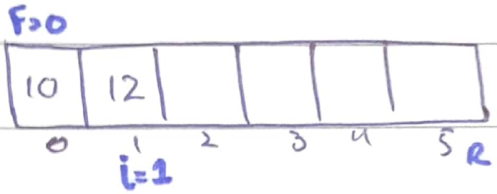
for condition will be processed again and again till the Queue is full

```
for( int i = F ;i<=R ;i++)
{
    cout << queue[i];
}
    cout << endl;
```

1 5

2

10 12



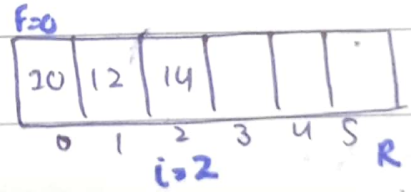
Again;

```
for( int i = F ;i<=R ;i++)
{
cout << queue[i];
    cout << endl;
}
```

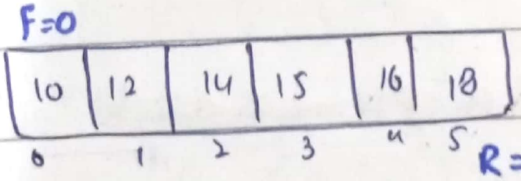
2 5

2

10 12 14



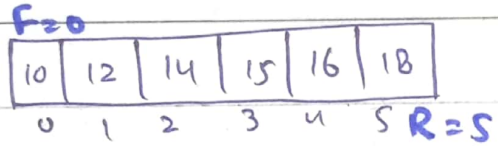
Repeat untill queue is full

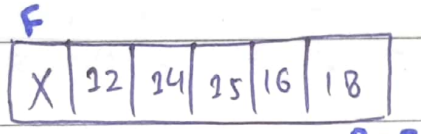


Output is;

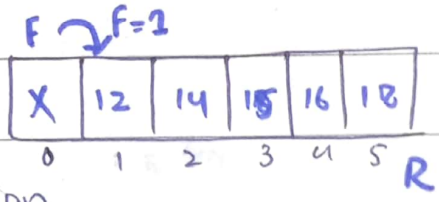
10 12 14 15 16 18

```
void delete()
{
        0   -1    0    5
  if(front==-1 || front>Rear)  → condition false
  {
    cout<<" Queue underflow";
  }
else
{
  cout<<" Element delete"<<queue[front]
```



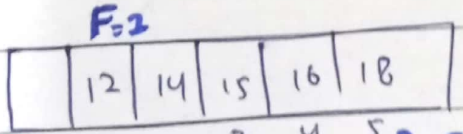


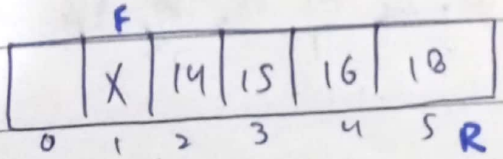

```
  front++;
}
```



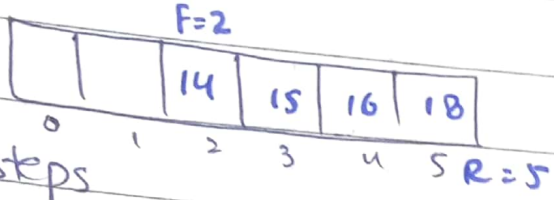
go back to if condition

```
  if(front==-1 || front>Rear)  → condition false
  {
    cout<<" Queue underflow";
  }
else
{
  cout<<" Element delete "<<queue[front];
```





front ++;



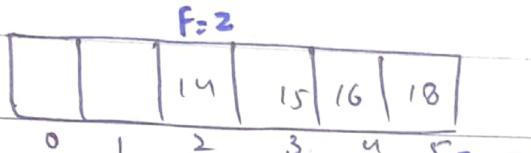
Repeat the steps

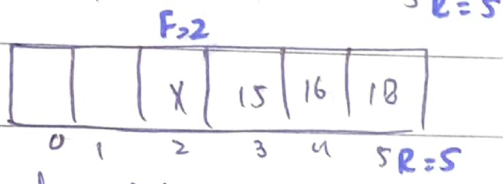
if(front == -1 || front > Rear) → condition false

{
cout << "Queue under flow";
}

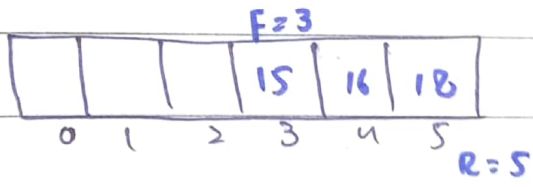
else

{ cout << "Element delete " << queue [front]

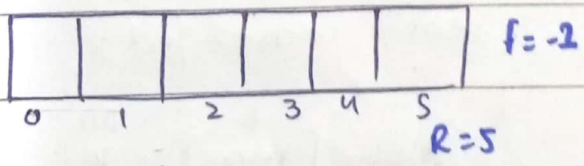




front ++;



Repeat Untill queue is empty



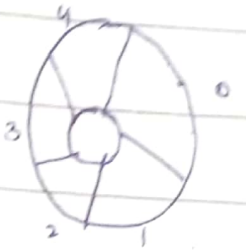
if(front==-1 || front > read) → condition True

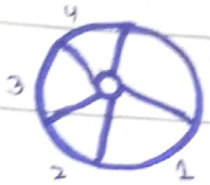
{
cout << "Queue underflow";
}

Queue underflow → print on screen

Circular Queue:

```
int queue[5] ; n = 5 ;
int front = -1 ;
int rear = -1 ;
```





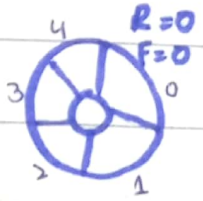
F = -2
R = -1

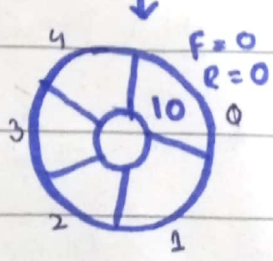
```
void insert ( int x )
{
    if ( R+1 %N == front )
    {
        cout << " Queue is full" ;
    }
    else
    {
        if ( front == -1 && Rear == -1 )
        {
            front = rear = 0 ;
        }
        else
        {
            rear = rear+1 %N ;
        }
        queue[rear] = x ;
 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

-1 +1 5 -1 → false move to else condition

-1 -1 -1 -1

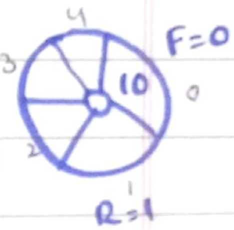
10 0 20

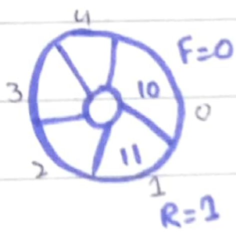




∴ repeat the function

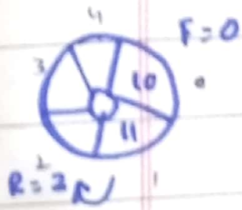
```
void insert (int x)
if ( rear+1 % N == front)  → false move to else part
{
    cout << " Queue is full";
}
else
{
    if ( front == -1 && rear == -1)  → false
    {
        front = rear = 0;
    }
    else
    {  rear = rear+1 % N;  →
    }
    queue[rear] = x;
```

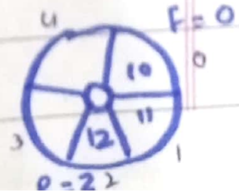




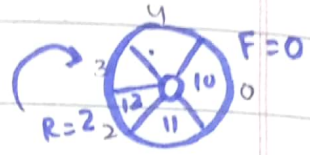
∴ again go back to function

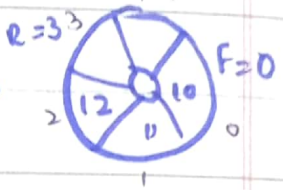
```
void insert (int x)
if ( rear +1 % N == front)  → false move to else condition
{
    cout << "Queue is full";
}
else
{
    if ( front == -1)          false
    { front = rear = 0
    }
    else { rear = rear +1 % N;
    }
    queue[rear] = x;
```

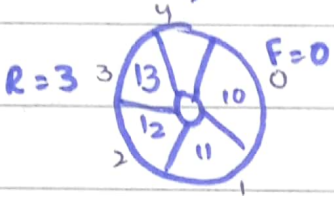




```
void insert (int x)
{
    if (rear+1 %N == front)  → false move to else part
    {
        cout<< "Queue is full";
    }
else
    {
        if(front == -1)  → false go to else part
        {
            front = rear = 0;
        }
        else
        {
            rear = rear+1 %N;
            queue[rear] = x;
        }
    }
}
```

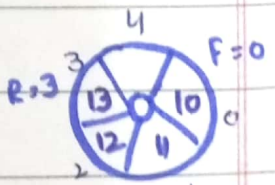


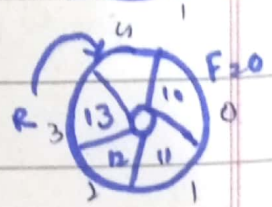


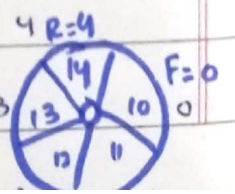


∴ again

```
void insert (int x)
{
    if (rear+1 %N == front)  → false go to else part
    {
        cout<< "Queue is full";
    }
else
    {
        if (front == -1)  → false
        {
            front=rear=0;
        }
        else
        {
            rear = rear+1 %N;
            queue[rear] = x;
        }
    }
```



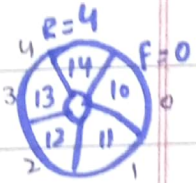


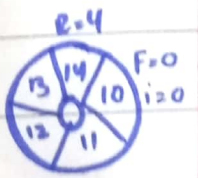


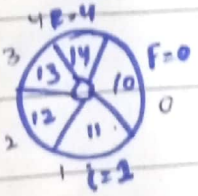
```
void insert( int x)
{
   if ( rear+1 %N == front)→ condition True
   {                          so skip all the
      cout << " Queue is full";   other condition.
   }
else {  if ( front == -1)
        {  front = rear = 0;
        }
   else {  rear = rear+1 % N;
        }
        que[rear] = x;
   }
}
```

"Queue is full" → display on screen.

```
void display ()
{
    int i = front
if (front == -1 && rear == -1) → false
   cout << "Queue is empty";
}
else
{  cout<< " Queue element";
   while( i != rear) → True
   {  cout << queue[i]; 10
      i = (i+1) % N;
   }  cout << queue[rear];
}
```

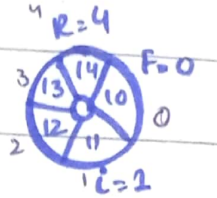




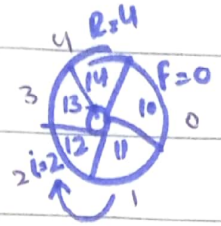


again call the processand move on till is condition is false

```
while( i != rear)
{
    cout << queue[i];
    i = (i+1) % N;
}
cout << queue[rear];
```



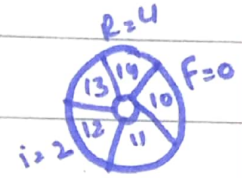
1 4 — 2 — 10 11 — 2 — 1+1 — 5

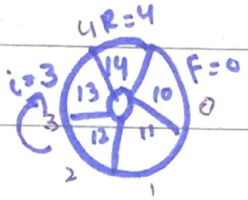


again run the while loop

```
while( i != rear) → True
{
    cout << queue[i];
    i = (i+1) % N;
}
cout << queue[rear];
```

2 4 — 2 — 10 11 12 — 3 — 2+1 — 5

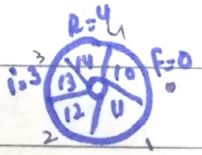


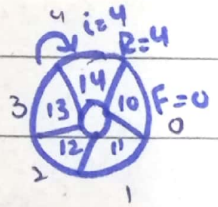


again

```
while( i != rear)
{
    cout << queue[i];
    i = i+1 % N;
}
cout << queue[rear];
```

3 4 — 3 — 10 11 12 13 — 4 — 3+1 — 5





again

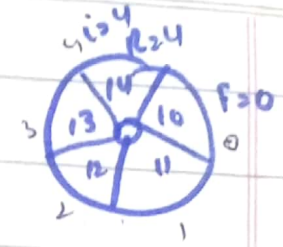
```
while (i != rear) → false
{
    cout << queue[i];
    i = i+1 % N;
```

4 4

}

cout << queue[rear];

10 11 12 13 14



output is;

10 12 12 13 14

void delete ( )

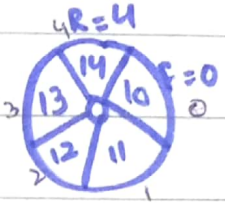
{

if (front == -1 && rear == -1) → false

{

cout << "empty";

}



else if (front == rear) → false

{

cout << "delete element" << queue[front];

front = rear = -1;

}

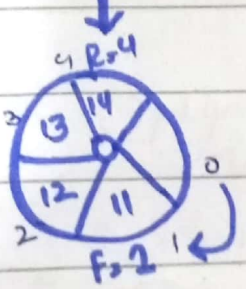
else

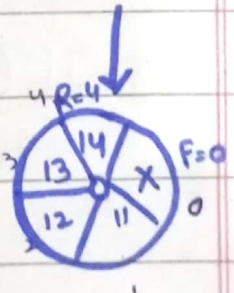
cout << "delete element" << queue[front];

front = (front + 1) % N;

}

}





∴ again call the process

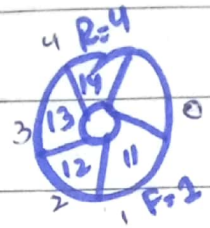
Day: Date:

if (front == -1 && rear == -1) → false

{

cout << " Empty ";

}



else

{

if(front == rear) → false

{

cout << " delete element " << queue[front];

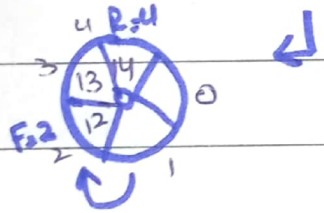
front = rear = -1;

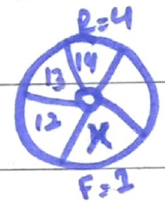
}

else {

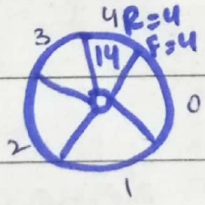
cout << " delete element " << queue[front];

front = front + 1 % N;





∴ repeat the process till the last index



if (front == -1 && rear == -1) → false

{

cout << "empty";

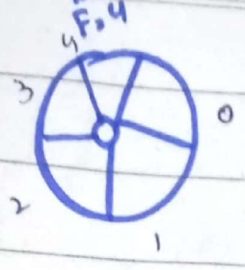
}

else

if (front == rear) → True

{

cout << "delete element" << queue[front];



front = rear = -1;

F = -1
R = -1

Go back to if condition

if (front == -1 && rear == -1) → True

{

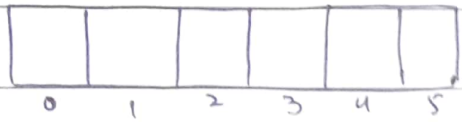
cout << " Empty ";

}

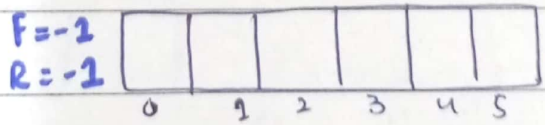
will print on screen.

Double ended Queue:

int deque[6], n = 6;



int front = -1, rear = -1



void insert front (int val)

{

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

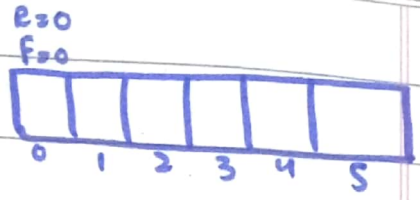
{ cout << " overflow ";

}

else

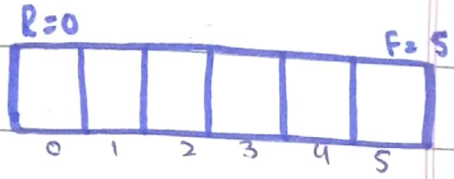
{ if (front == -1) → -1, -1

front = rear = 0;



else if (front == 0) → 0, 0

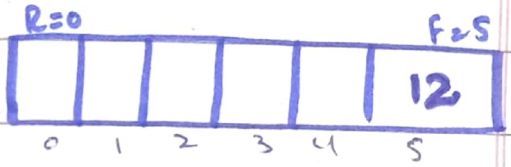
front = n-1; → 6-1



else → skip this part

front--;

deque[front] = val; → 5, 12



∴ go back to function

void insertfront(int val) → 13

{ if (front == 0 && rear == n-1) → 5, 0, 5 → false ~~|| (front = rear+1)~~

{ cout << "overflow";

}

else

{ if (front == -1) → 5 → false

front = rear = 0;

}

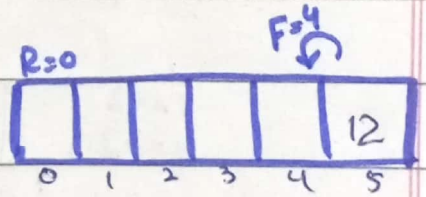
else if (front == 0) → 5 → false go to else part

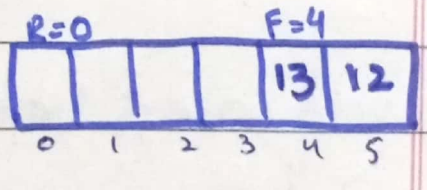
front = n-1;

else

front--;

deque[front] = val; → 4, 13



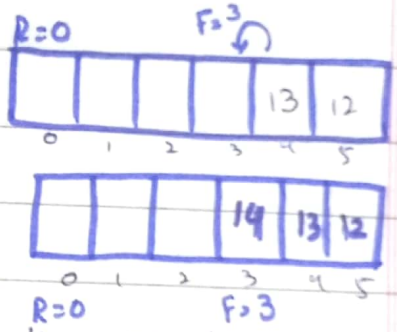


∴ again back to function

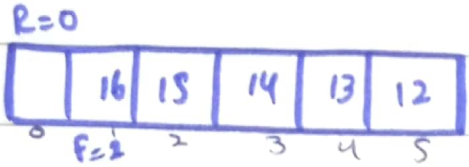
void insertfront(int val) → 14

{ if (front == 0 && rear == n-1) → 4, 0, 5 → false

{
cout<< overflow;
}
else
{
if(front == -1) → false
front = rear = 0;
else if (front == 0) → false
front = n-1;
else
front--;
deque [front] = val;



∴ repeat the process till the east index

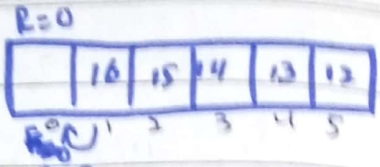


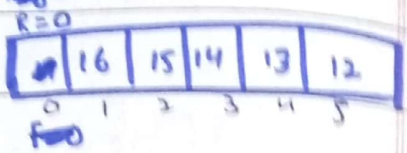
void insert(int val)
{
if(front==0 && rear = n-1) → false
{
cout<< "overflow";
}
else
if(front == -1) → false
front = rear = 0;
else if(front == 0) → false
front = n-1;

else

front--;

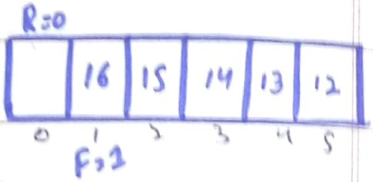
deque[front] = val;





∴ again call the function

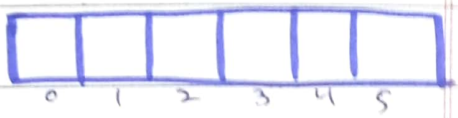
```
void insertfront(int val)
{
    if((front == 0 && rear == n-1) || (front = rear+1))
    {
        cout<< "overflow";
    }
```

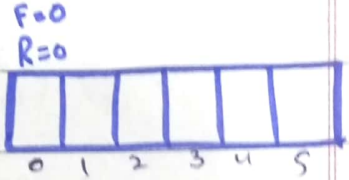


"overflow" → will print on screen

void insertrear(int val)

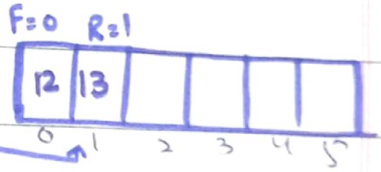
```
{
    if((front==0 && rear==n-1) || (front = rear+1))
    {
        cout<<"overflow";
    }
else
    if(rear == -1) → True
    {
        front = rear = 0;
    }
else if(rear==n-1)
    rear = 0;
else
    rear++;
    deque[rear] = val;
}
```



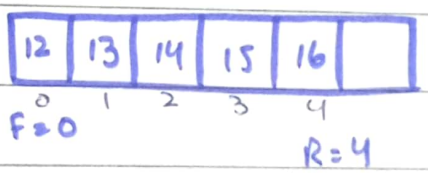


∴ go back to function

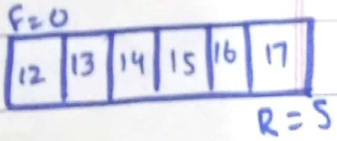
```
void insertrear(int val)                    val = 13
if( front==0 && rear==n-1 || (front == rear+1) → false
                                  0      5
{
    cout<<"overflow";
}
else
    if (rear==-1) → false  go to else part
                  0
    {
        front = rear = 0;
else if( rear = n-1 ) → false
                  0   5
        rear = 0;
else
        rear ++;
        deque[rear] = val;
```



∴ repeat the process

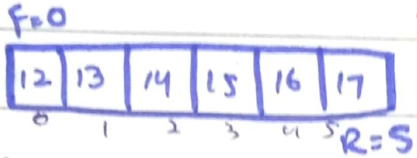



```
void insert rear( int val)                  val = 17
if( front==0 && rear==n-1 || (front == rear+1) → false
                  0     4   5
{
    cout<<"overflow";
}
else
    if(rear==-1) → false
                  4
    {
        front = rear = 0;
    }
else if ( rear = n-1) → false
                  4   5
        rear = 0;
else
        rear ++;
                                  17
        deque[rear] = val;
```



Day: Date:

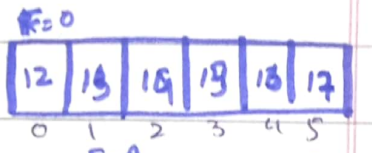
```
void insertrear (int val)
if( front==0 && rear==n-1) || (front=rear+1)
{                                  True
  cout<< "overflow";
  }
```

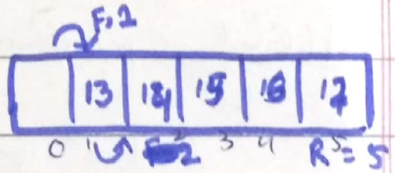


Skip all the conditions.

"Overflow" → print on screen.

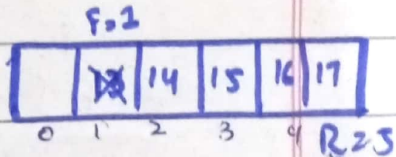
```
void delete front ( )
{
  if (front==-1) →false
  { cout<<"underflow";
  }
else
    cout<< deque[front]
  if (front==rear) →false
      front=rear=-1;
else if( front==n-1) →false
      front=0;
  else
      front++;
```





∴ repeat procedure

```
  if (front==-1)
    { cout<<"underflow";
  else
      cout<< deque[front]
  if (front==rear)
      front=rear=-1;
```

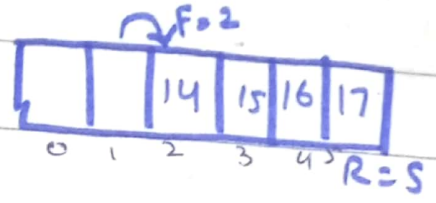


else if (front == n-1)

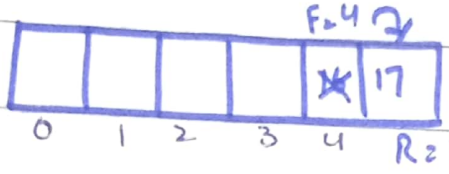
front = 0;

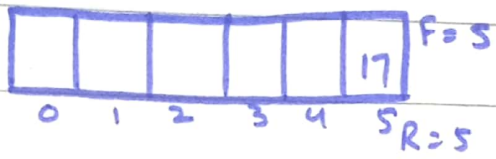
else

front ++;



∴ repeat process





if (front == -1)

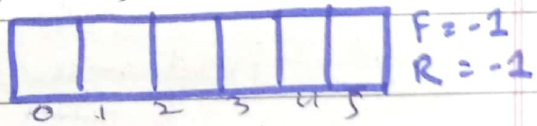
{ cout<< "underflow";

}

else if (front == rear) → True

front = rear = -1;

cout<< deque[front];



"underflow" → print on screen.

void deleterear()

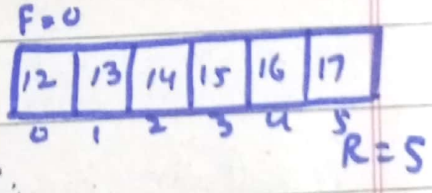
{

if(rear == -1) → false

{ cout<< "underflow";

}

else {



Day:

cout<< "Deque element" << dque[rear(5)] <<endl;

~~if(rear == front)~~

if(front(0) == rear(5)) → false

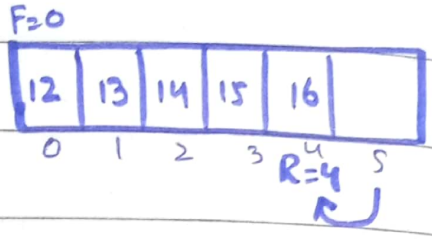
front = rear = -1;

else if(rear(5) == 0) → false

rear = n-1;

else

rear --;



∴ repeat the process

if(rear(4) == -1) → false

{

cout<< "underflow";

}

else {

cout<< "deque element" << dque[rear(4)] <<endl;

if(front(0) == rear(4)) → false

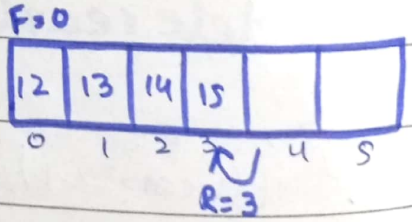
front = rear = -1;

else if(rear(4) == 0) → false

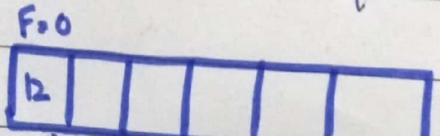
rear = n-1;

else

rear --;



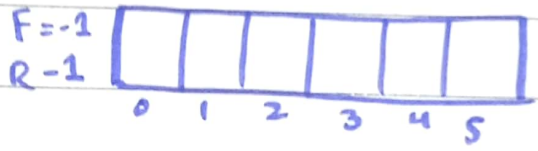
repeat the process till the queue is empty



Day:

Date:

```
if (rear == -1)
{   cout<< "underflow";
}
else
{ cout << "deque element" << deque[rear];
  if ( front == rear) → True
        front = rear = -1}
```

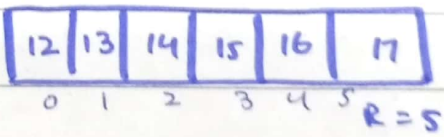


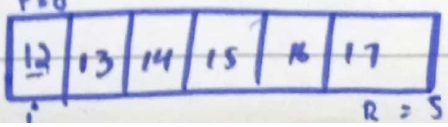
∴ again

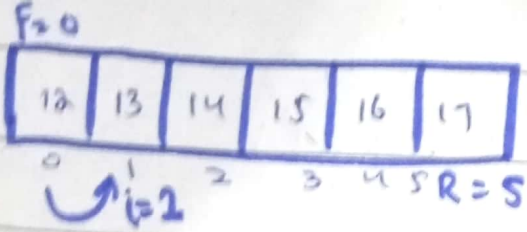
```
if (rear == -1) → True
{
  cout<< "underflow";
}
```

print on screen

```
void display ()
{
if (front == -1) → false
{
 cout<< " Empty ";
}
else
{  cout<< " Deque elements are ";
   if ( rear > = front) → True → skip the else part
   {   for ( int i = front; i <= rear; i++)
            cout<<deque[i] << "  "; 12
   }
```







∴ repeat the process

if (front == -1) → false

{ cout << "Deque elements are"

}

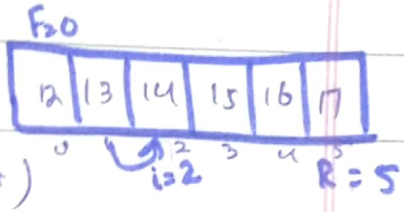
else

if (rear >= front) → True

{ for (int i = front; i <= rear; i++)

cout << deque[i] << " "; 12 13

}



else

for (int i = front; i < n; i++)

cout << deque[i] << " ";

for (int i = 0; i <= rear; i++)

cout << deque[i] << " ";

}

}

}

∴ repeat untill the last index

output is;

12 13 14 15 16 17