"Lab Assignment No.2"

Submitted by:-

Mubeshra Shafique

Roll no.:- SP22-BCS-120

Section:- B

Subject:-

Data Structure (Lab)

Submitted to:-

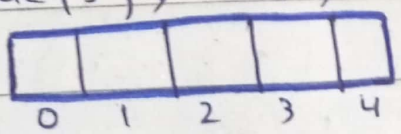
Mam Yasmeen Jana

"QUEUE"

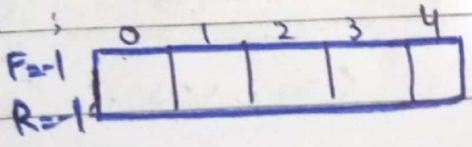
Types of Queue:-

1- Linear Queue:-

① int queue[5], n=5;



int F=-1, R=-1;



→ void enqueue (){

int val;

declare the variable

Date: / / Day: Mon Tue

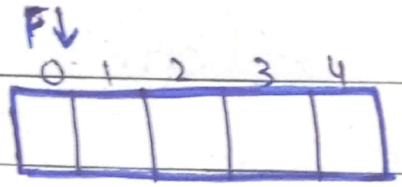
if (R == n-1) → False so, skip
(-1, 5-1=4)

cout<< "Queue Overflow" <<endl;

else {

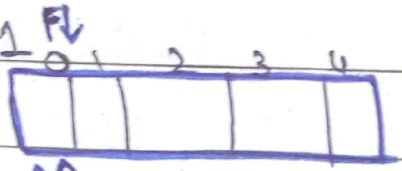
if (F == -1) → True
(-1 == -1)

F = 0;

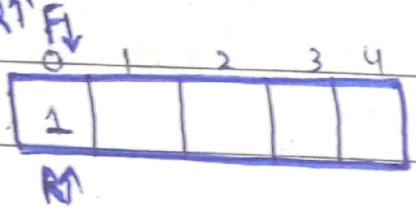


Cin>> val; 1

R++;



queue[R] = val;



② again call the enqueue function to insert the element in next index.

int val;

if (R == n-1) → False
(0 == 4)

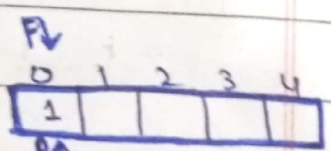
cout<< "Queue overflow" <<endl;

skip move to else part

else {

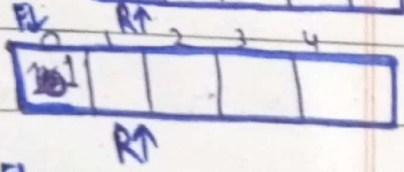
if (F == -1) → False
(0 == -1)

F = 0; skip

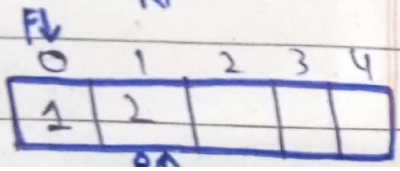


Cin>> val; 2

R++;



queue[R] = val;



③ again insert the value in 2nd index

if (R == n-1) → False

Cout << "Queue overflow" << endl;

else {

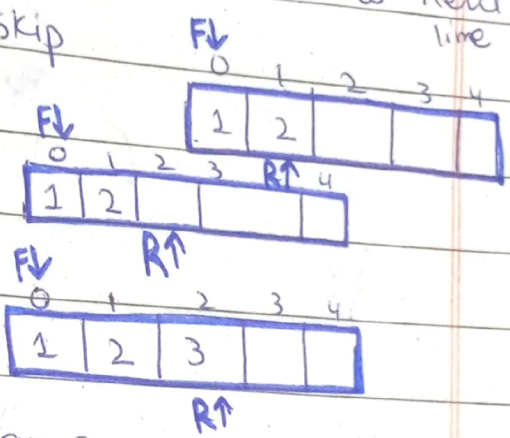
if (F == -1) → False move to next line

F = 0; skip

cin >> val; 3

R++;

queue[R] = val;



④ So, on the same process till the last index

if (R == n-1) → False

Cout << "Queue Overflow" << endl;

else {

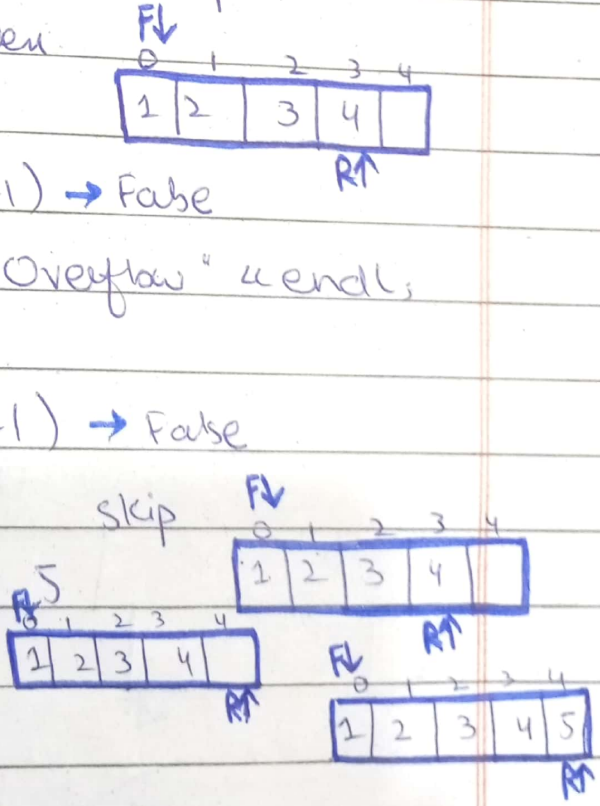
if (F == -1) → False

F = 0; skip

cin >> val; 5

R++;

queue[R] = val



⑤ again insert

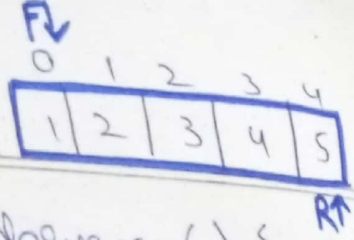
if (R == n-1) → True

Cout << "Queue Overflow" << endl;

Prints on the screen.

Date: / /

Day:



→ void dequeue () {

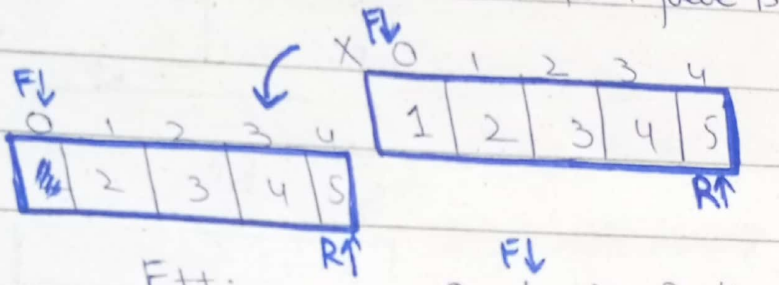
① if (F == -1 || F > R) { → False

cout << " Queue underflow " ;

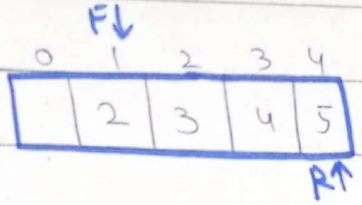
}

else {

cout << " Element deleted from queue is " << queue[F];



F++;



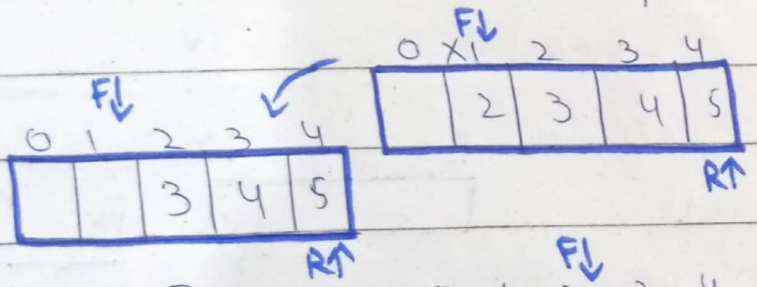
② again dequeue the second element.

if (F == -1 || F > R) → False

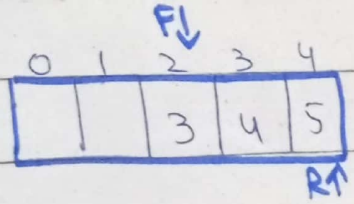
cout << " Queue underflow " ;

else {

cout << " Element deleted from queue is " << queue[F];

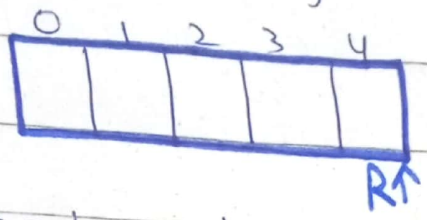


F++;



③ So, on the same process till

the queue is empty



-1 ==-1 → 4

if (F==-1 || F>R) → TRUE

cout<<" Queue underflow";

prints on the screen.

→ void display(){

if (F==-1)

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

else{

cout<<"Queue elements are: ";

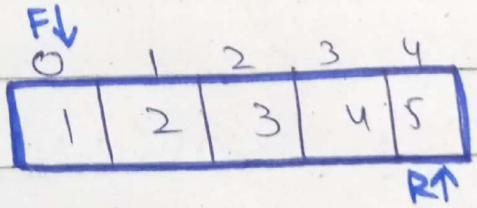
for(int i=F; i<=R; i++){

cout<<queue[i]<<" ";

cout<<endl;

}

}



0 ==-1

if (F==-1) → false

else{

① for(int i=F; i<=R; i++){

0 0<=4

cout<<queue[i]<<" ";

0

cout<<endl; → 1

② for(int i=F; i<=R; i++){
cout<< queue[i] << " "; → 2
cout<< endl; }

③ for(int i=F; i<=R; i++){
cout<< queue[i] << " "; → 3
cout<< endl;
}

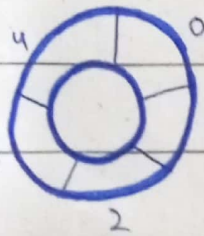
④ for(int i=F; i<=R; i++){
cout<< queue[i] << " "; → 4
cout<< endl;
}

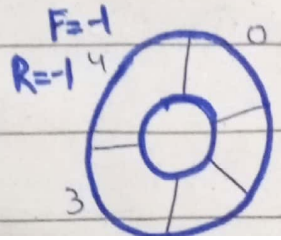
⑤ for(int i=F; i<=R; i++){
cout<< queue[i] << " "; → 5
cout<< endl;

output:
1
2
3
4
5

2- Circular Queue

int queue[5], N=5;





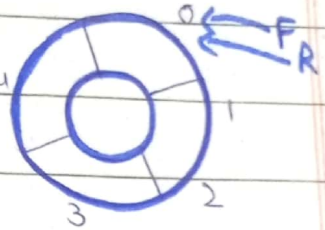
int F=-1, R=-1;

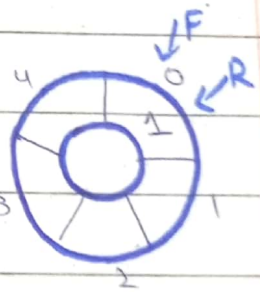
Day: Mon Tue Wed Thu Fri Sat

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

(-1+1) % 5 == 0 == -1 False

TRUE

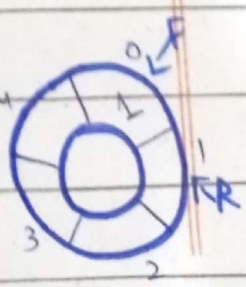




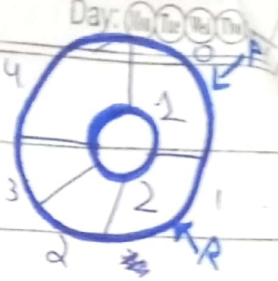
② void enqueue (int x) {
if ((R+1) % N == F) {
cout << " Queue is Full ";
}
else {
if (F == -1 && R == -1) {
F = R = 0 ;
}
else {
R = (R+1) % N ;
}

0+1 % 5 == 0 == 0 False

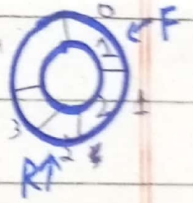
False so, SKIP this

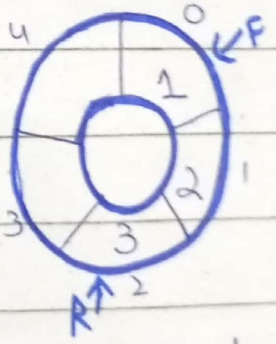


queue[R] = x;
}}



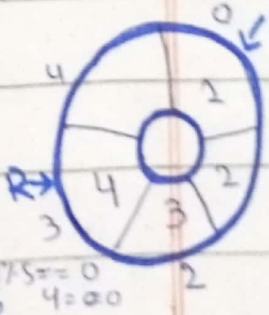
③ void enqueue(int x){
if((R+1)%N == F){ → 2%5 == 0, 2==0, False
cout "Queue is full";
}
else{
if(F==-1 && R==-1){
F=R=0; → False
}
else{
R = (R+1)%N; → 2%5=2
}
queue[R] = x;
}}





④ same process till the last index.

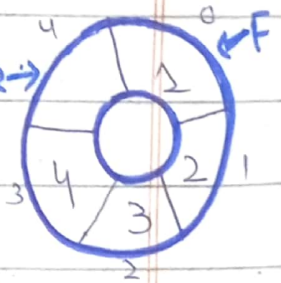
void enqueue(int x){
if((R+1)%N == F){ → 4%5 == 0, 4==0, False
cout "Queue is full";
}

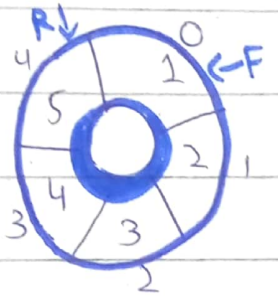


Day: Mon Tue Wed Thu Fri Sat

```
if ( F==-1 && R==-1){
    F=R=0;          → False
}
else{
    R = (R+1) % N;
}
queue[R] = x;
}
}
```

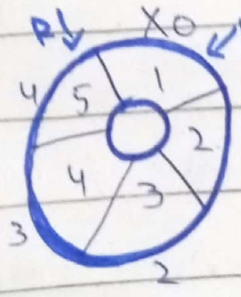
3+1 % 5
4 % 5 = 4





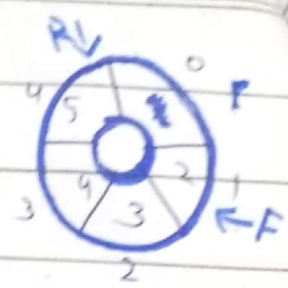
→ void dequeue(){

```
if ( F==-1 && R==-1){   → False
    cout<<" Queue is empty";
}
else if ( F == R){      → False
cout<<" delete element"<<queue[F];
    F=R=-1;
}
else {
cout<<"dequeue element"<<queue[F];
~~F=(F+1)%N;~~
```



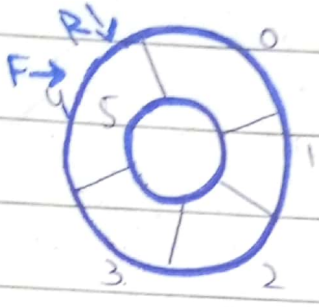
F = (F + 1) % N

}



② Void dequeue (){

if and else if false till the last index.



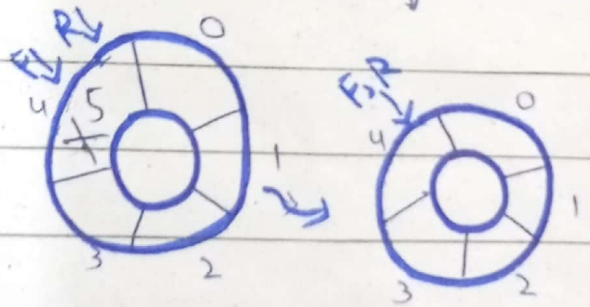
if (F == -1 && R == -1){

cout << "Queue is empty"; → False

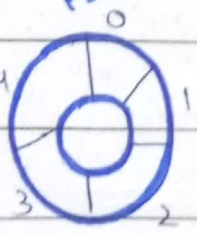
}

else if (F == R) { → True

cout << "dequeue element" << queue[F];



F = R = -1



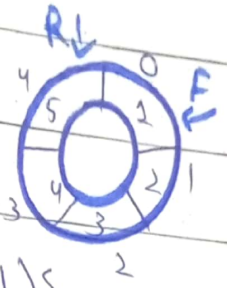
③ if (F == -1 && R == -1)

cout << "Queue is empty";

print on screen

→ void display () {

int i = F;



if (F == -1 && R == -1){

cout << " Queue is empty ";

}

else {

cout << " Queue elements ";

while (i != R){

cout << queue [i] << " ";

i = (i+1) % N;

}

cout << queue [R] << endl;

}

}

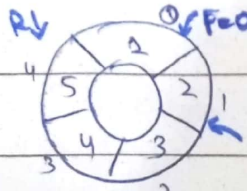
int i = F; F=0

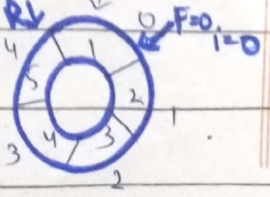
Queue is not empty more on else part

while (i != R) { True

cout << queue [i] << " ";

i = (i+1) % N;



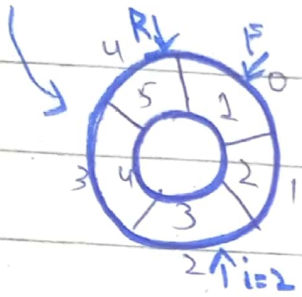


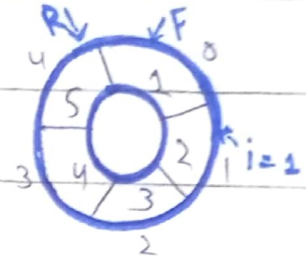
move on while till the condition is false.

1 ! = 4
while (i ! = R) { → True
1

Cout<< queue[i] << " "; → 2

2 ↙ 1+1 %5 2%5=2
i = (i+1) %N

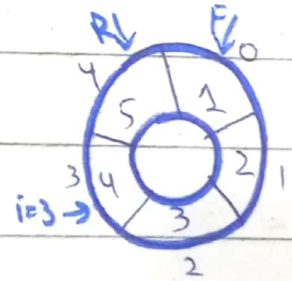


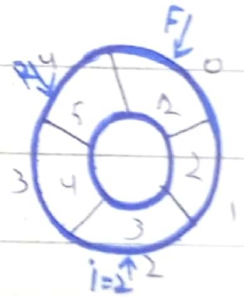


2 ! = 4
while (i ! = R) { → True
2

Cout<< queue[i] << " "; → 3

3 ↙ 2+1 %5 3%5=3
i = (i+1) %N

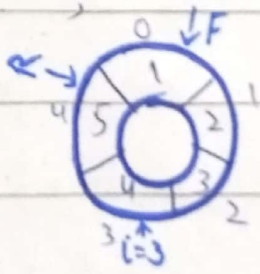


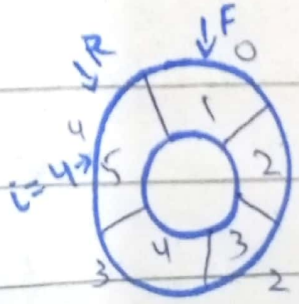


3 ! = 4
while (i ! = R) { → True
3

Cout<< queue[i] << " "; → 4

4 ↙ 3+1 %5 4%5=4
i = (i+1) %N





Day: Mon Tue Wed Thu Fri Sat

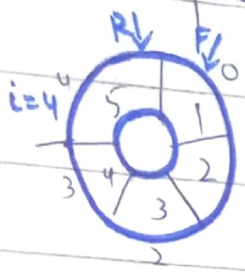
4 != 4

while (i != R) { → False

}

cout << queue [R] << endl.

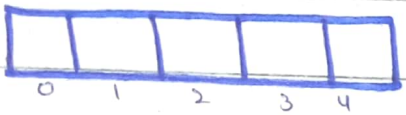
Skip this part



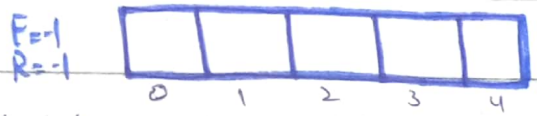
↳ 5

Double ended Queue:-

int deque[5], N=5



int F=-1, R=-1;



void insert front (int x) {

-1 -1+1=0

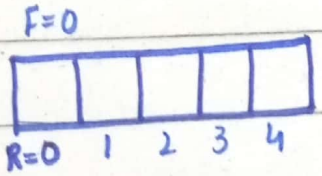
if ((F == 0 && R == N-1) || (F == R+1)) {

cout << "Queue is overflow"; False

}

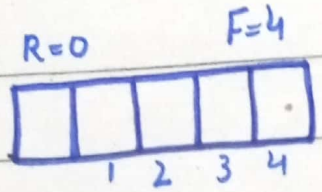
else {

if (front == -1)



front = rear = 0;

else if (front == 0)

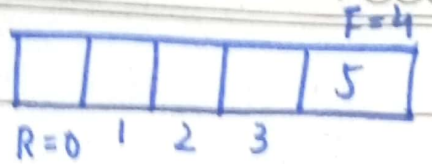


front = n-1;

∵ skip else part

deque [front] = val;

Day:



void insert front (int val)

∵ go back to if condition

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

{ → false go to else part

cout << "overflow";

}

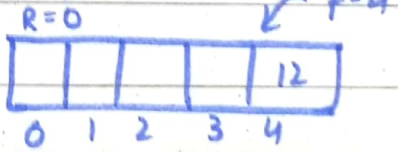
else {

if (front == -1) { → false

front = rear = 0; }

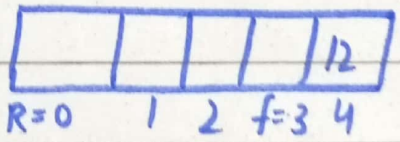
else if (front == 0) → false

front = n-1;

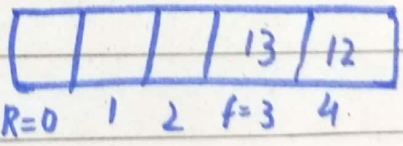


else

front--; →



deque [front] = val;



void insert front (int val) → 14

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

{

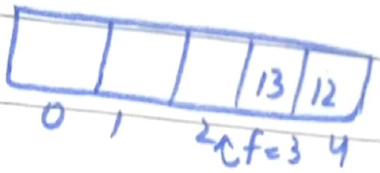
cout << "overflow"; → false

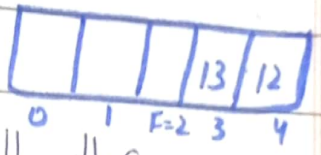
}

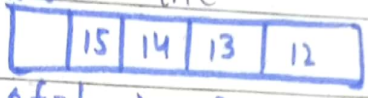
else {

if (front == -1) {

front = rear = 0;
else if (front == 0)
front = n-1;
else
front--;
deque[front] = val;
Repeat the process till the

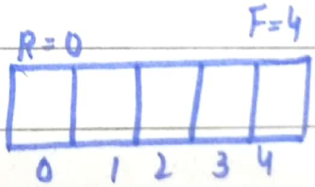




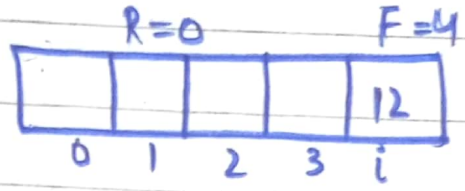


void insert_front(int val) → 16
if (front == 0 && rear == n-1) || (front == rear+1)
{ cout << "overflow";
}
→ True
∴ Now print overflow on the screen

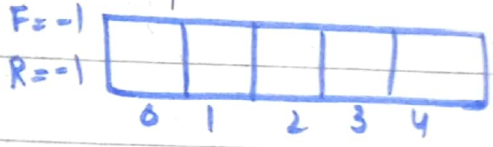
void display() {
if (front == -1)
{ cout << "empty";
}
else {
if (rear >= front) { → false
for (int i = front; i <= rear; i++)
cout << deque[i];
}
else {



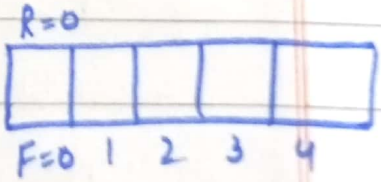
```
for (int i= front ; i < n ; i++)
    cout << deque [i];          → 12
for (int i=0 ; i <= rear ; i++)
        cout << deque [i];
}}
```

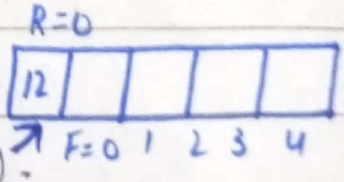


again repeat else part




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





again repeat the insertRear

function

void insert Rear (int val) → 13

if (front == 0 && Rear = n-1) || (front == Rear + 1)
0 0 0 4 0 1

{ → false

cout << "overflow" ; }.

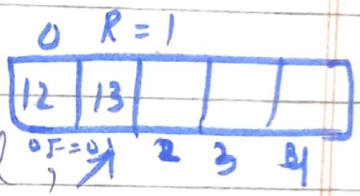
else

if (rear == -1) {

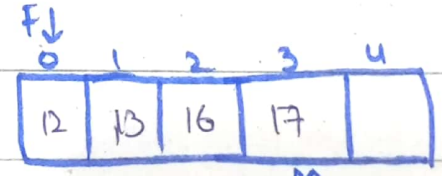
front = rear = 0 ;

else rear ++ ;

deque [rear] = val ;



Repeat this process till the last index



void insert rear (int x) {
18

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

{ → false

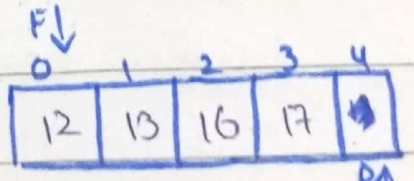
cout << "Queue is Overflow" ;

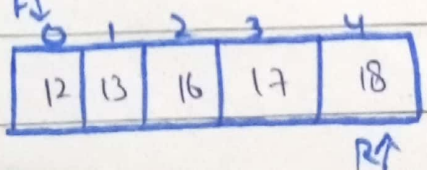
}

else {

rear ++ ;

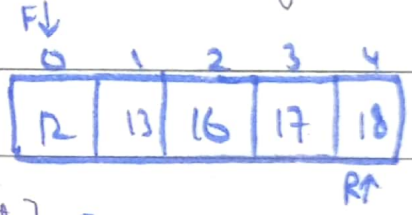
deque[rear] = x ;
18

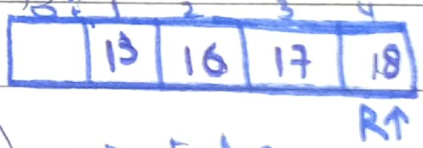


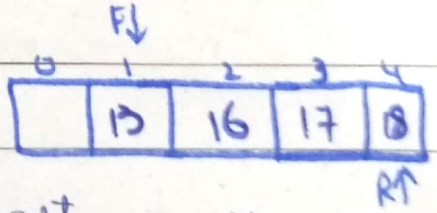


go back to if condition

```
if ((front == 0 && rear == n-1) || (front = rear+1))
    cout << "Queue is Overflow";        → True
}
void delete front (){
  if ( front == -1) {
  cout << " Queue is Underflow";
  }
  else {
  cout << deque[front];
  if ( front = = rear)                  → False
    front = rear = -1;
  else if ( front = n-1)                → False
      front = 0;
  else
      front ++;
```

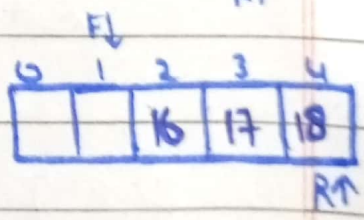


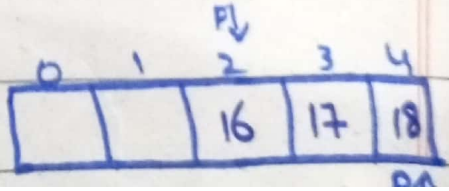




Repeat else part

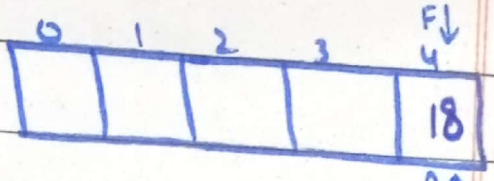
```
else {
  cout << deque[front];
    if (    ) skip
    else if ( ) skip
else
    front ++;
```



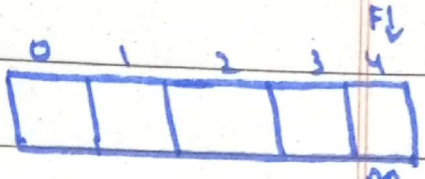


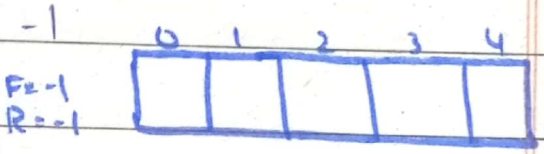
Day: Mon Tue Wed Thu Fri Sat

Repeat this process till the last index




```
if (front == -1) {        → False
cout<< " Queue is underflow";
    }
else {
cout<< deque [Front];
  if (front == rear)
  front = rear = -1
  }
```

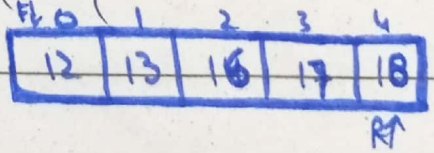




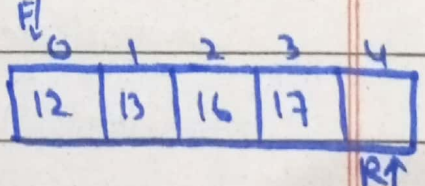
again

```
  if (front == -1) {      → True
cout<< " Queue is underflow";
    } print on the screen.
```

```
void deleteREAR( ){
```




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



Date: / /

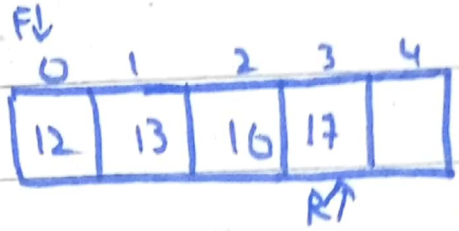
4 == 0

else if (rear == 0) → False

rear = n-1;

else

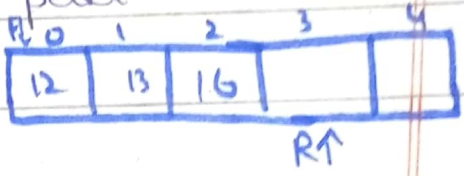
rear--;



}

go back to else part

else {



cout << deque [rear];

if (front == rear) → False

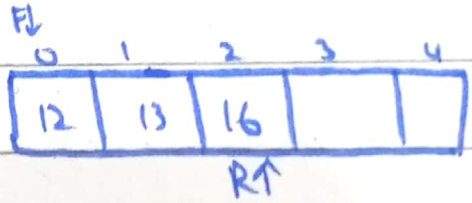
front = rear = -1;

else if (rear == 0) → False

rear = n-1;

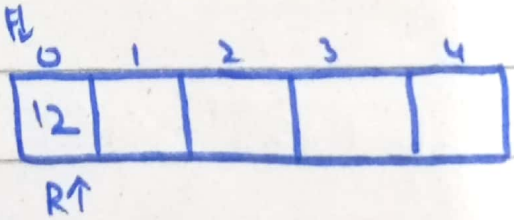
else

rear--;



}

→ Doing the same process till the last index.

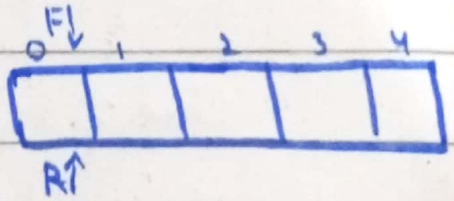


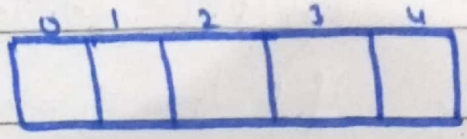
else {

cout << deque [rear];

if (front == rear)

front = rear = -1





else if (rear == 0) → False

rear = n-1;

else → False / Skip

rear --;

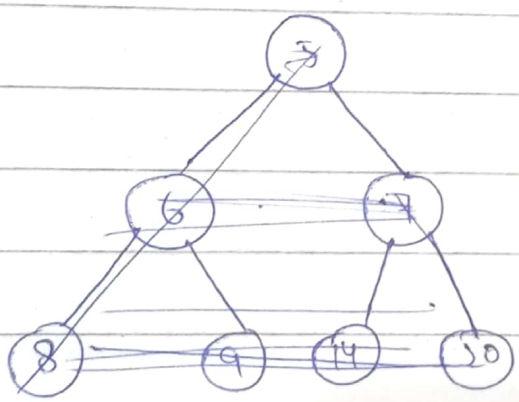
again

if (rear == -1) { → TRUE

cout << "Queue is underflow";

} print on the screen

Priority Queue:-



for(int i=0 ; i<7 ; i++){

if (i==0) {

cin >> arr[i]; → 5

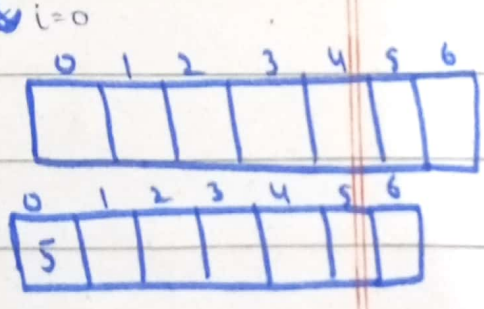
size = i;

}

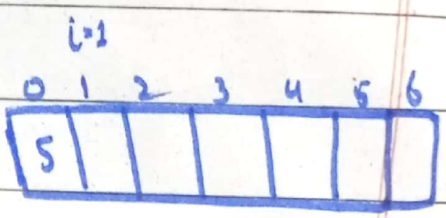
else {

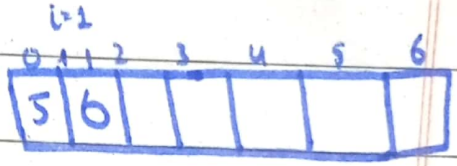
cin>> arr[i];

size = i; }



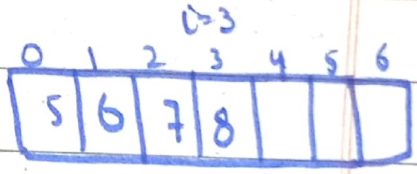
```
}
for( int i=0; i<7; i++){
    if (i==0){ → False skip
        cin>> arr[i];
        size = i;
    }
else{
    cin>>arr[i];
    size= i;
    }
}
```

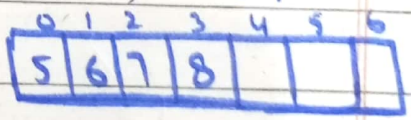


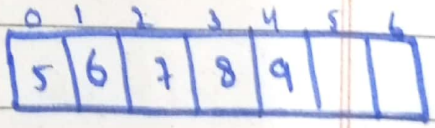


same process

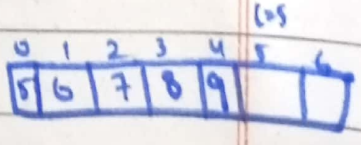
```
for( int i=0; i<7; i++){
    if (i==0){ → False
        cin>> arr[i]; size = i;
    }
else{
    cin>> arr[i];
        size = i;
    }
}
```



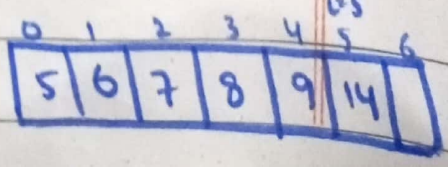





```
    size = i;
    }
}
```




```
for (int i=0; i<7; i++){
    else {
        cin>> arr[i];
        size= i;
```



```
for( int i=0; i<7; i++){
    else{
        cin>> arr[i];
    }
}
```

0	1	2	3	4	5	6 (i=6)
5	6	7	8	9	14	20