Date:

Name: Sadia Javed

Roll No: sp22-BCS-113

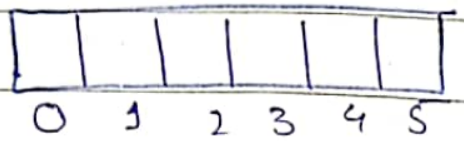
Section: BCS-B

Subject: Data Structure and Algorithm LAB

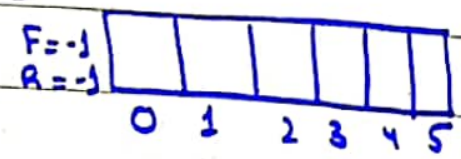
Assignment NO 3

. Linear Queue:

int queue[6], n = 6



int front = -1 , Rear = -1



void insert(){

int value (Declare a variable)

if (Rear = n-1) condition wrong goes to else Part

{

cout<< "Queue overflow";

}

else

if (front == -1) condition true goes to next line.

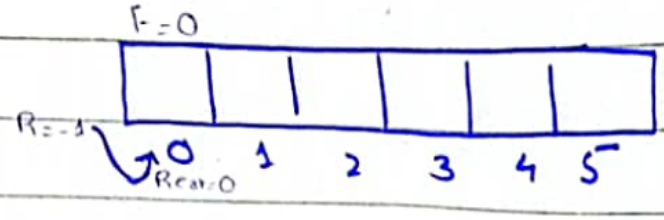
front = 0

else

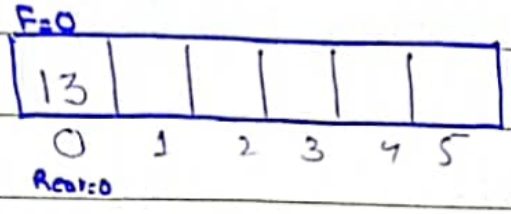
cin >> value; (Take input from users

Day: _____ Date: _____

rear++;



queue[rear] = value;



Again go back to if condition:

if (rear == n - 1)
(0 == 5)

{

cout << "Queue overflow";

}

else

if (front == -1) It is false so

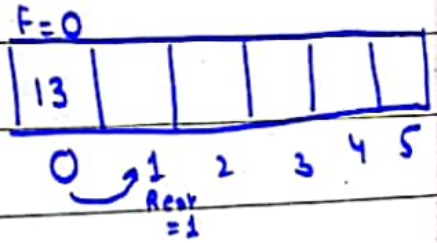
front = 0 front = 0

else

14

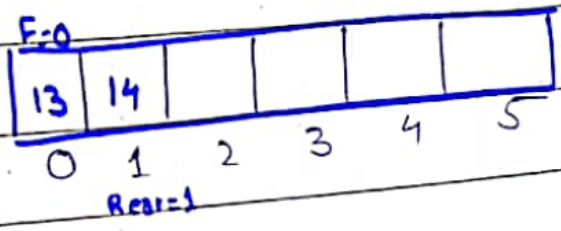
Cin >> value; F=0

Rear ++;



1 14

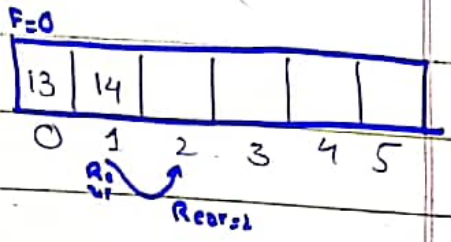
queue[Rear] = value

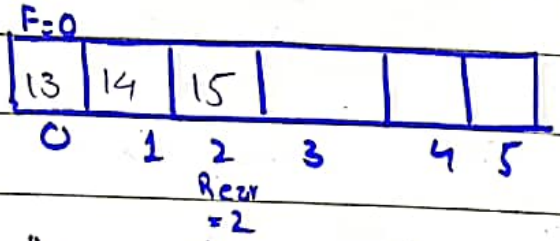


Date:

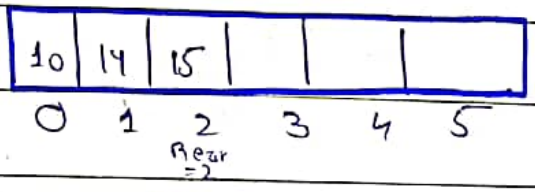
Now again go back to if condition:

```
if( Rear = n - 1)
{
    cout << " Queue overflow";
}
else
    if( front = = -1)   → condition false
        front = 0;
    else
        cin >> value;
        Rear++;
        queue[Rear] = value
```





Repeat these steps until end;



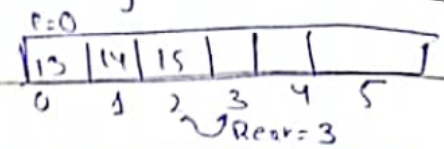

```
if(rear = = n - 1)   → condition false
{                      go back to else part
    cout << " Queue overflow";
}
else
    if( front = = -1)   condition false
```

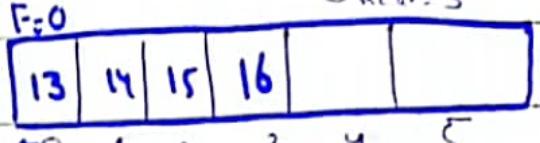
Day: Date:

front = 0; go to next line

cin >> value; (15)

Rear++;





queue[Rear] = value; (3, 16) Rear → Rear = 3

Now again go back to if condition

if (Rear == n-1) (3, 5) → condition false go back to else part

{

cout << "Queue overflow";

}

else

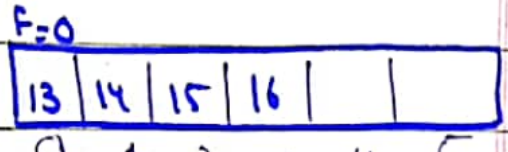
if (front == -1) (0, -1) condition false

front = 0; go to next line

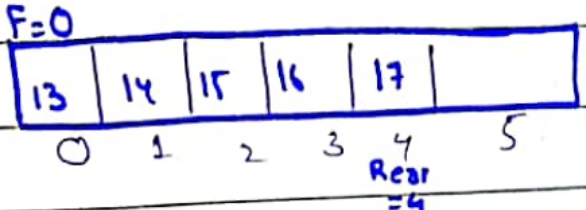
else e.

cin >> value; (17)

Rear++;

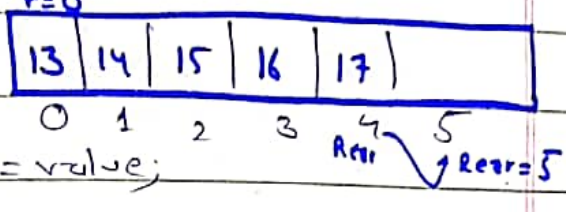


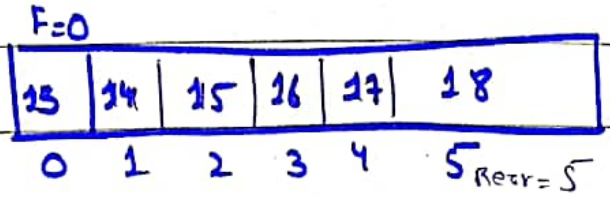
queue[Rear] = value; (4, 17)



Now again go back to if condition

```
if (Rear == n-1)   -> condition false go
{                     to else part
   cout<< "Queue overflow";
}
else
   if( front == -1)   condition false
      front = 0;      go to next line
else
   cin>>value;
   Rear++;
   queue[Rear] = value;
```





Now, go back to if condition.

if (Rear = n-1) -> True so else

{ condition will not run

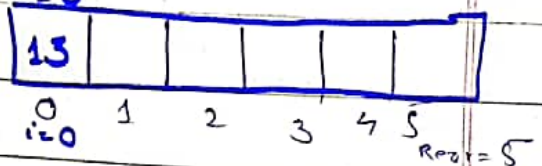
cout<< "Queue overflow";

}

Now the "Queue" is full skip all other conditions.

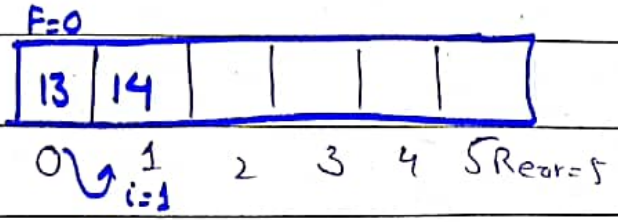
"Queue overflow" → print on screen

```
void display(){
  if(Front == -1)        -> condition false go
  {                         to else part
    cout << "Queue is empty";
  }
  else
    cout << "Queue elements are";
    for(int i = front; i < Rear; i++)
      cout << queue[i];
      cout << endl;
```



For condition will be processed again and again till the Queue is full.

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



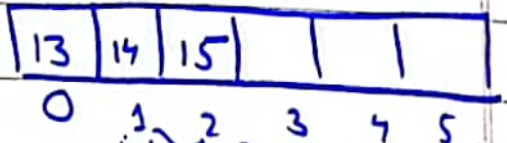
Now again repeat for condition;

```
for(int i = front; i <= Rear; i++)
```

Date:

3
cout << queue [i];

cout << endl;



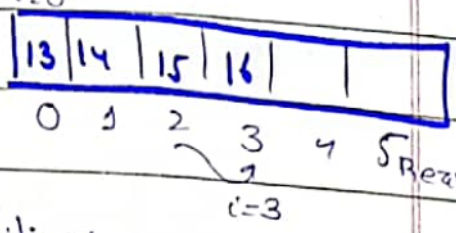
Repeat req min for condition;

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

{

3
cout << queue [i]

cout << endl;



}

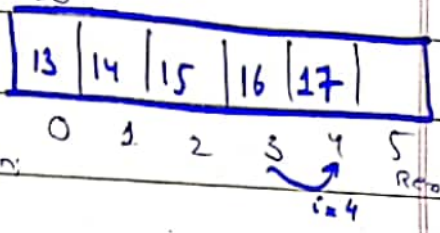
Repeat req min for condition;

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

{

4
cout << queue [i];

cout << endl;



}

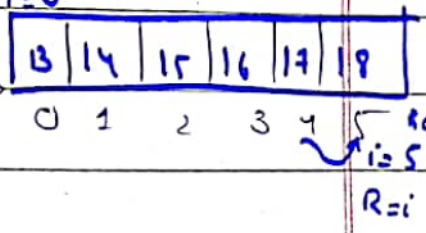
Repeat req min for condition;

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

{

5
cout << queue [i];

cout << endl;

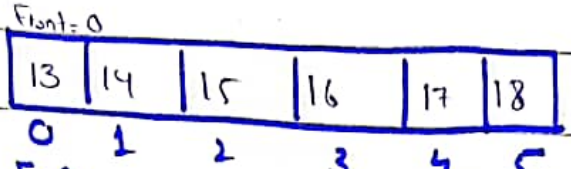


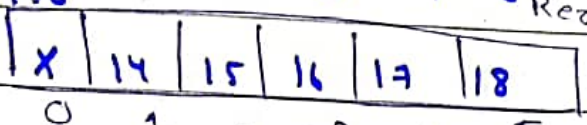
Now is queue is full;

Output is; 13, 14, 15, 16, 17, 18

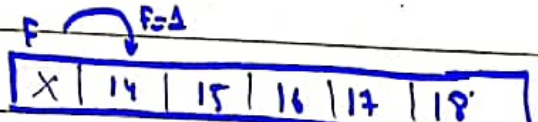
13 14 15 16 17 18

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





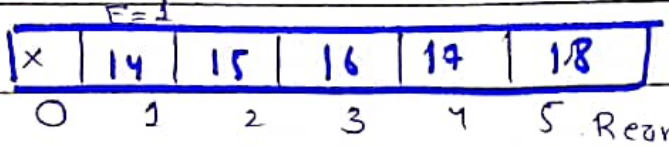
front++;

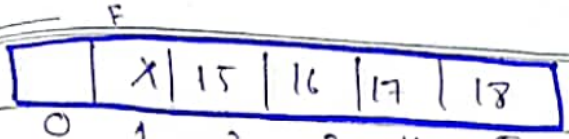


Go Back to if condition;

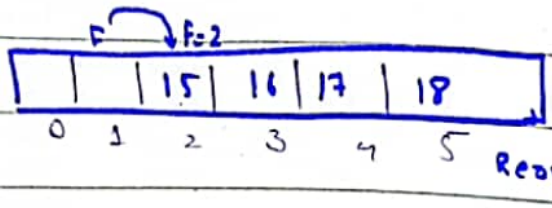
if(front == -1 || front > Rear) -> condition false

```
  {
    cout<<" Queue underflow";
  }
else
  {
    cout<<" Element delete"<<queue[front];
```





Front ++;



Now again Repeat the steps;

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

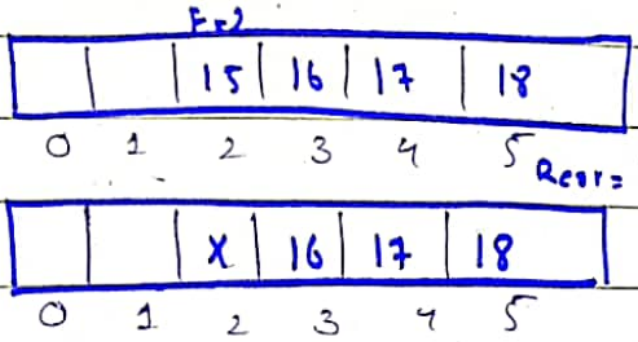
{

cout << " Queue underflow ";

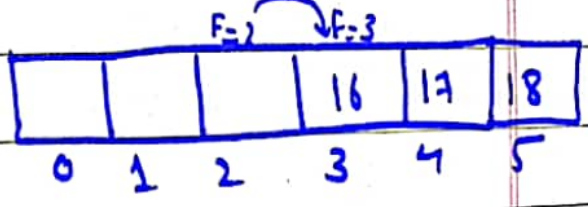
}

else {

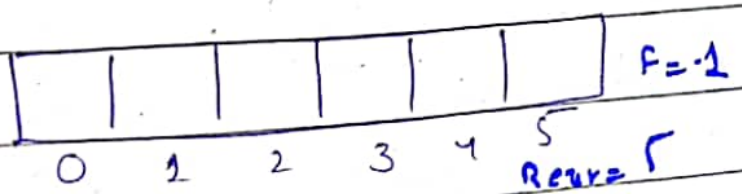
cout << " Elements deleted " << queue [Front]



front ++;



Repeat until queue is empty;



Day: Date:

if (front == -1 || front > rear) → condition true

{

cout << " Queue underflow ";

}

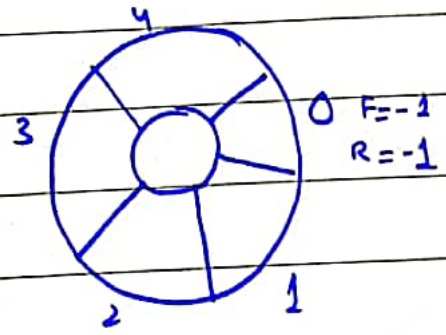
Queue underflow → print on screen

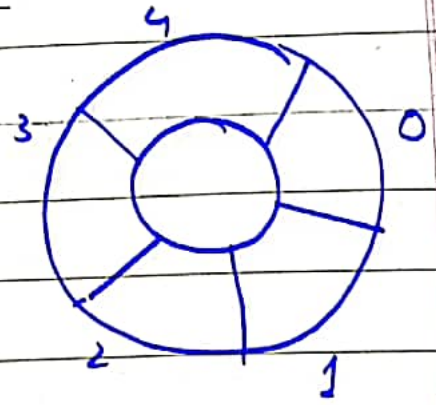
(2)

Circular Queue:

int queue [5], N = 5

int front = -1, Rear = -1





void insert (int N) {

if ((R + 1) % N == front) → false

(-1 + 1) (5)

{ cout << " Queue is full ";

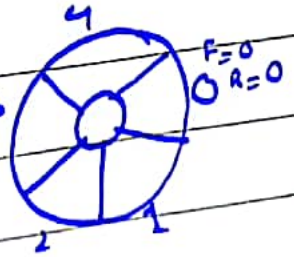
}

else {

if ((front == -1 && Rear == -1) → true

(-1) (-1) (-1) (-1)

front = Rear = 0



else {

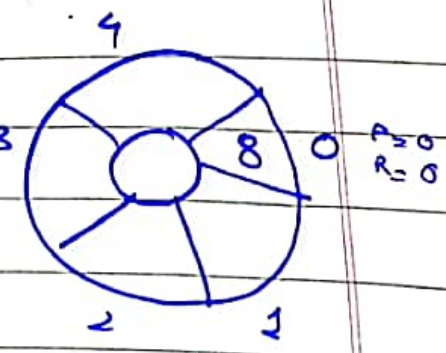
Rear = ((Rear + 1) % N;

}

Queue [Rear] = x;

}

}



Now again repeat the funtion;

void insert (int x)

if((Rear+1) % N == front){ → false

cout << "Queue is full";

}

else {

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

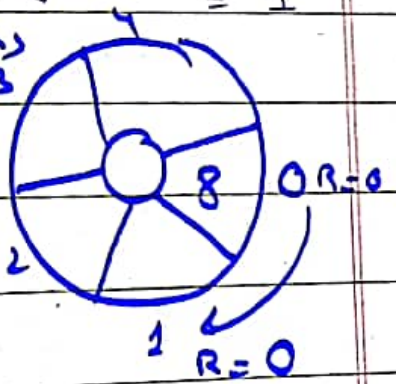
front = Rear = 0;

}

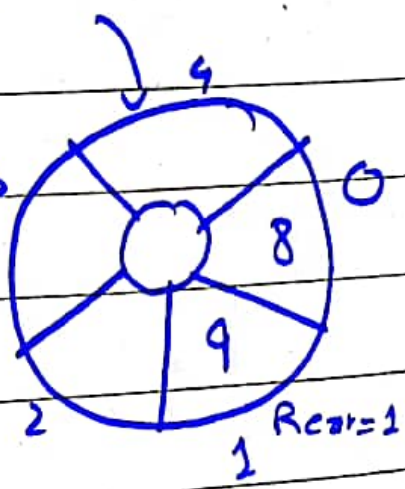
else {

Rear = (Rear+1) % N

}



Queue [Rear] = x



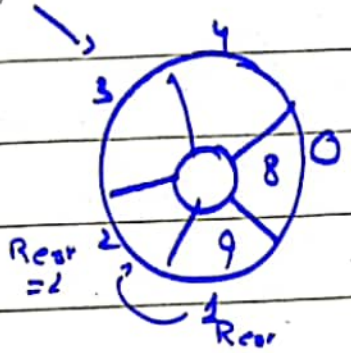
Day:

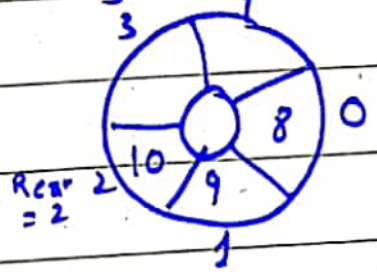
Again repeat the function

```
void insert(int 10 u){
if((R+1)%5 == front){ -> false
cout << " Queue is full ";
}
else
if(front == -1 && Rear == -1) -> false
{ front = Rear = 0;
}
else {
Rear = (Rear+1)%N
}
Queue[Rear] = u
}
```

1+1 % 5 → 2

Rear = 2





The same procedure is applied till the last index.

```
void insert(int 11 u)
if((Rear+1)%5 == front) -> false
{
cout << " Queue is full ";
}
```

2+1 % 5 → 3 ≠ 0

else

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

{

front = Rear = 0;

}

else {

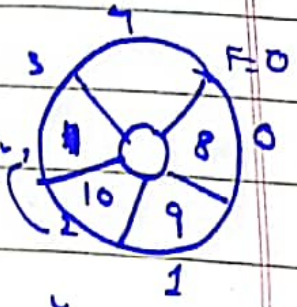
Rear = (Rear + 1) % N;

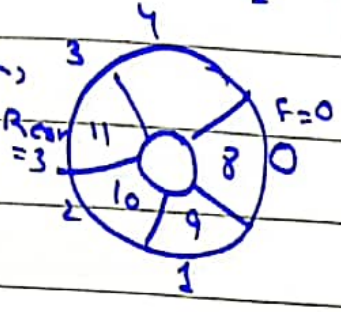
}

Queue[Rear] = x;

}

}





Again repeat the steps

void insert (int x)

if((Rear+1) % N == front) → false

{

cout << " Queue is full";

}

else

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

{

front = Rear = 0;

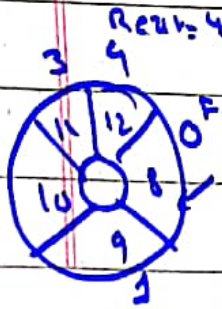
}

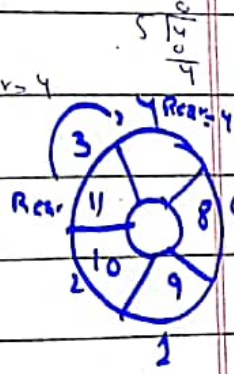
else {

Rear = (Rear + 1) % N;

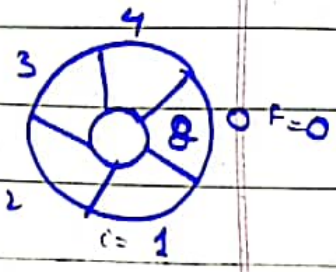
}

Queue[Rear] = x





```
void display(){
  int i = front;
  if (front == -1 && Rear == -1) -> false
  {
    cout << "Queue elements";
    while (i(0) != Rear(4)){ -> true
      cout << queue[i];
      i = (i+1) % N(5);   ->
```



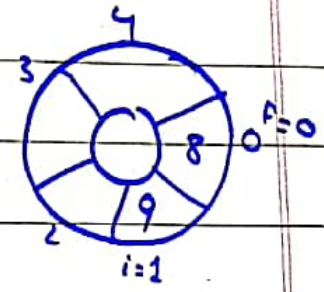
Rear = 5 → 0 (5 % 5 = 0)

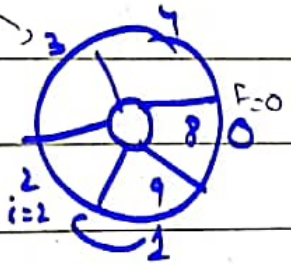
Now, move on till the condition is false;

while (i(1) != Rear(4)) -> true

cout << queue[i(1)] -> 9

i = (i+1) % N(5) i = 1+1 = 2 Rear = i = 2

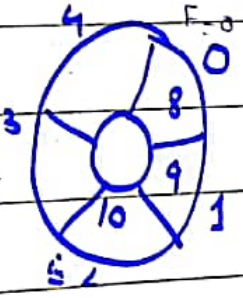


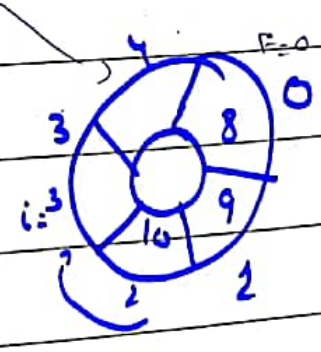


while (i(2) != Rear(4)) -> true

cout << queue[i(2)] -> 10

i = (i+1) % N i = 2+1 = 3





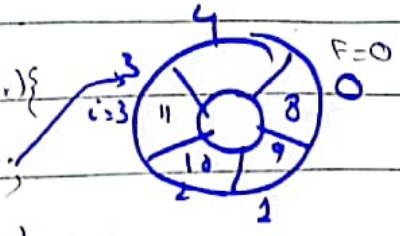
Date:

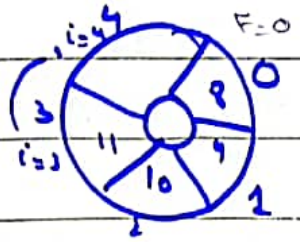
Repeat this step till it reaches the last index

while(i != Rear){

cout << queue[i];

i = (i+1) % N ; i = 4





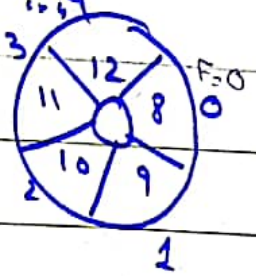
Now again repeat;

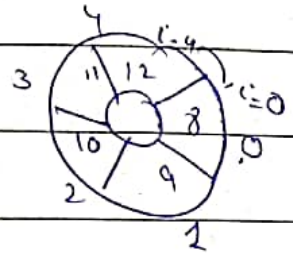
while(i != Rear){

cout << queue[i]; → 12

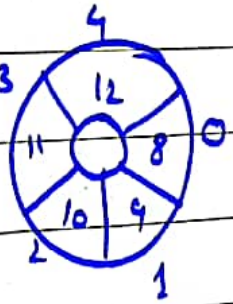
i = (i+1) % N = 0

5 % 5 = 0





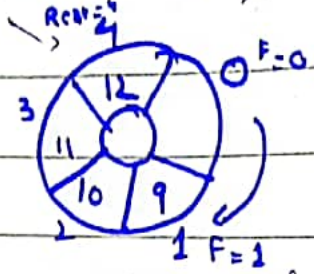

```
void delete(){
    if( front == -1 && Rear == -1){ -> false
        cout << " Queue empty";
    }
    else if( front == 0 == Rear) -> false
    {
        cout << " delete element" << queue[front]
        front = Rear = -1;
    }
```

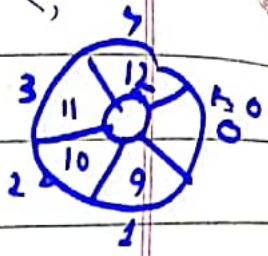


else

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

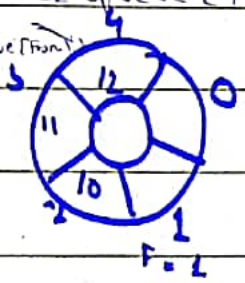
front = (front + 1) % N

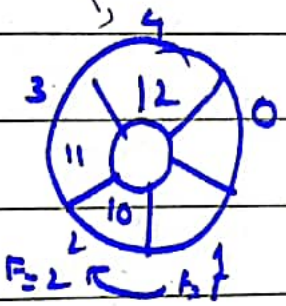




Now, again repeat the function;

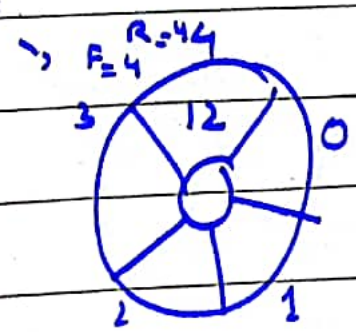
```
void delete(){
if(front==-1 && Rear==-1) -> false
{
    cout << "Queue empty";
}
else
    if(front==Rear)
    {
        cout << "delete element" << queue[front];
        front = rear = -1;
    }
else {
    cout << "delete element" << queue[front];
    front = (front + 1) % N;
```





Repeat the if condition until it reaches the last index.

if



Date:

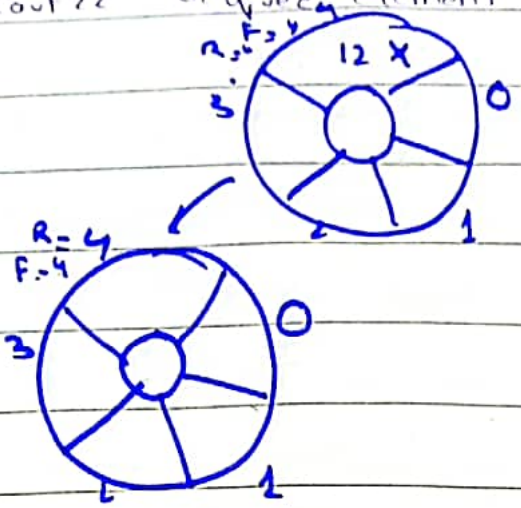
if (front == -1 && Rear == -1){

cout << " Queue is Empty ";

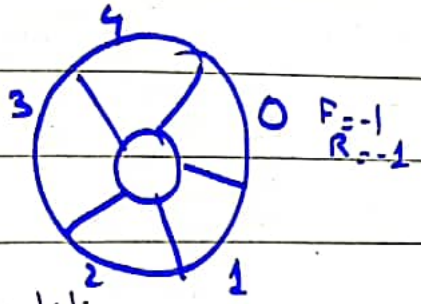
}

else if (front == Rear) -> true

{ cout << " dequeue element " << queue[Front];



Front = Rear = -1



Now go to if condition

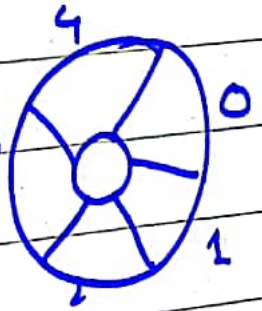
if (front == -1 && Rear == -1) -> true

{

cout << Queue is Empty ";

}

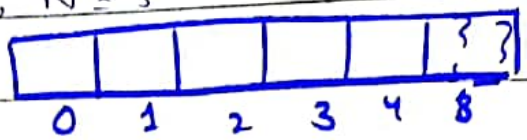
print on screen



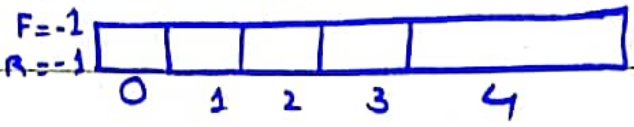
3

Double ended Queue:

int deque[5], N = 5



int front = -1 ; Rear = -1



void insert front (int val){

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

{ cout << "overflow"; → false

}

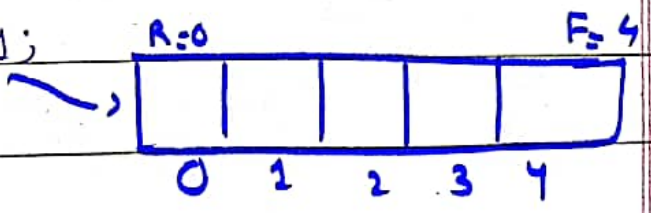
else {

if (front == -1) → true

front = rear = 0;

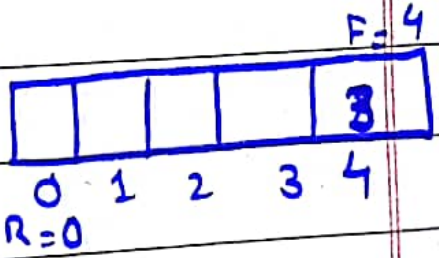
else if (front == 0)

front = n-1;



∴ Skip else part

deque [Front] = val;



Date:

void insert front (int val)

Now go back to if condition

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

{

cout << "Queue overflow";

}

else {

if (front == -1){

front = rear = 0;

} else if (front == 0)

front = n-1;

else

front--;

deque[Front] = val;

Now again go back to function.

void insert front (int val)

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

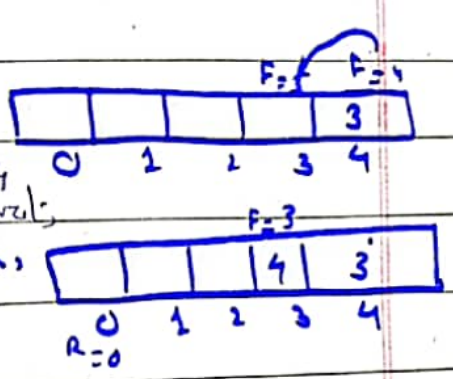
{

cout << "Overflow";

}

else {

if (front == -1){



Day:

front = rear = 0;

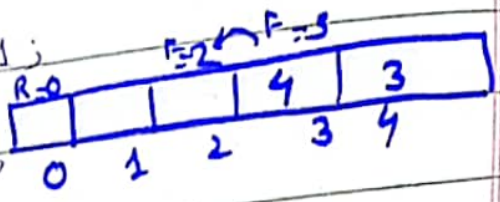
else if (front == 0) → false

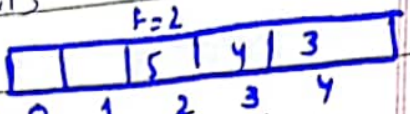
front = n-1;

else

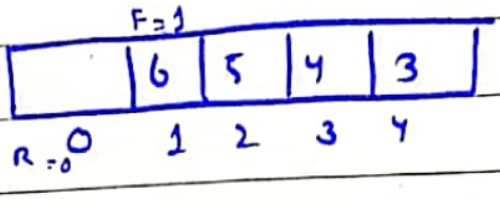
front--;

deque[front] = val;





Repeat the process till i reaches 1.



Now again go back to function;

void insertfront(int val)

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

{

cout << "Overflow";

}

Now print overflow on screen

void insertRear(int val){

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

{ cout << "overflow";

}

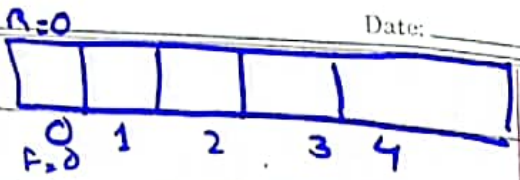
else

if (rear == -1) {

Date:

front = Rear = 0;

}



else if (rear = n-1)

rear = 0;

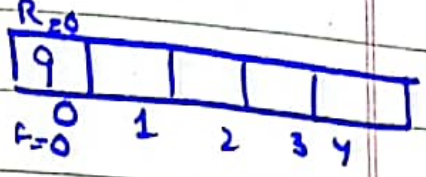
else

{

rear ++;

}

deque [rear] = val;



}

Again repeat the function.

void insertRear (int val)

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

→ false

{

cout << " Queue overflow";

}

else {

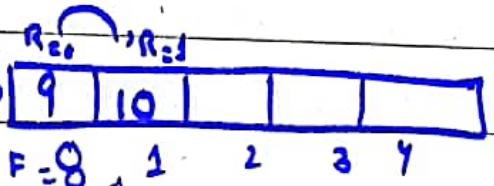
if (rear == -1) {

front = rear = 0 → false

else

{

rear ++; →

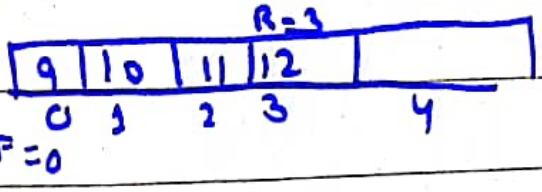


deque [rear] = val

}

}

Repeat this process till the last index



Void insertrear(int val) -> 13

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

{

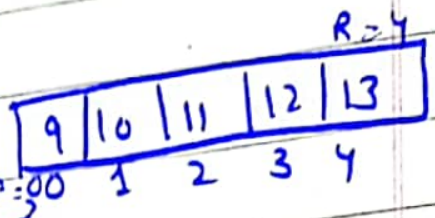
cout << "Overflow"; -> false

}

else

rear++; ->

deque[rear] = val;



Now go back to if condition,

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

{

cout << "overflow"; -> True

Print on screen

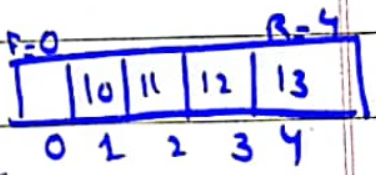
void deletefront(){

if (front == -1)

{ cout << "underflow";

}

else



cout << deque[front]

if(front == Rear)

front = rear = -1

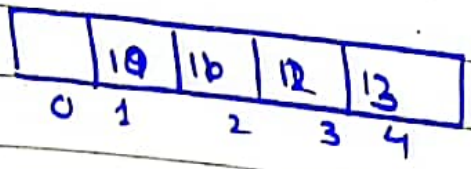
else if (front = n-1)

front = 0;

else

Date:

front ++;



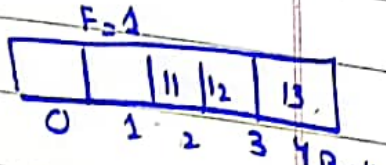
}

}

Again repeat the else part

else

cout << deque [front] ;



if (front == Rear)

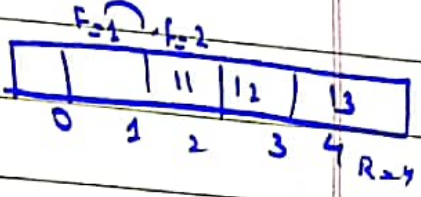
front = rear = -1;

else if (front == n-1)

front = 0;

else

front ++;



Repeat this process till the last index.

4 -1

if (front == -1) {

cout << " underflow ";

}

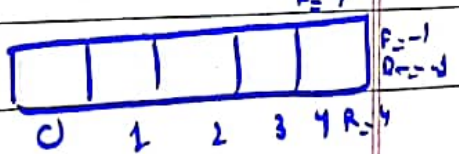
else

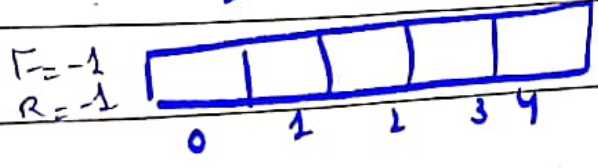
4

cout << deque [front]

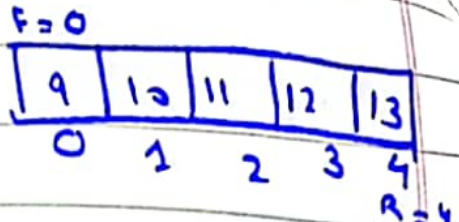
if (front+1 == Rear4)

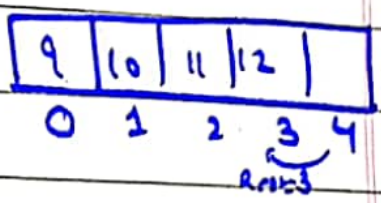
front = rear = -1





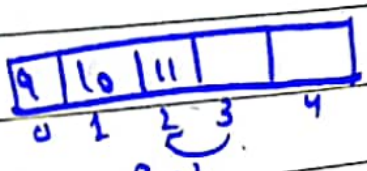
```
void deleteRear(){
if(rear == -1)          -> false
{ cout << "underflow";
}
else
    cout << deque[rear];
    if(front == rear)   -> false
        front = rear = -1
    else if (rear == 0) -> false
        rear = n-1;
    else
        rear--;
}
```





Now 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--;         ->
}
```

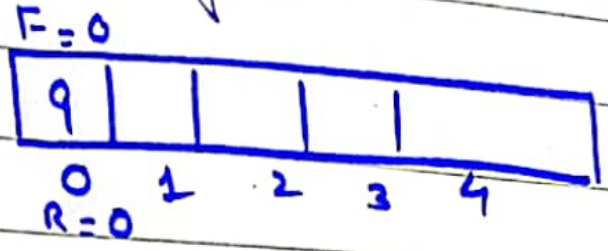


Now go back to else - part

else

cout << deque [rear]

Repeat the process untill the queue is empty.

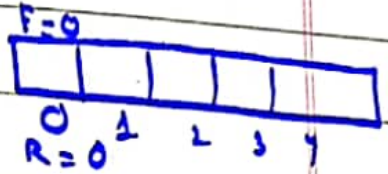


else

cout << deque [rear]

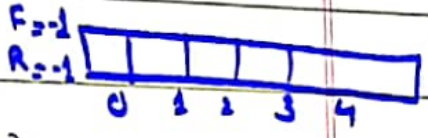
if (front == Rear)

front = Rear = -1



Now;

go to if part



if (rear == -1)

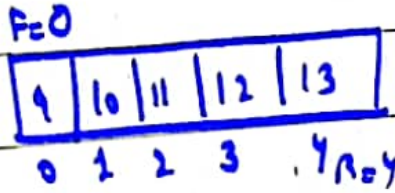
{

cout << "underflow";

}

Print on Screen

void display()



{

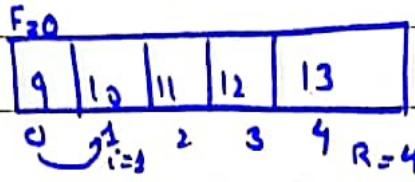
if (front == -1) -> false

{

cout << "Empty";

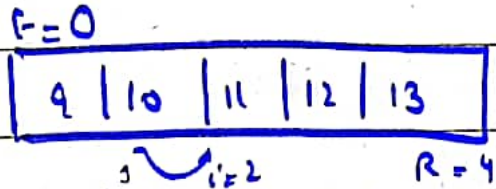
Day:

```
else
{
cout << " Deque elements are ";
if(rear >= front) -> True
{ for(int i= front; i<= rear; i++)
cout << deque[i] << " ";
}
```



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 until the last index

output is;

9 10 11 12 13