Page No.:
Date: / /

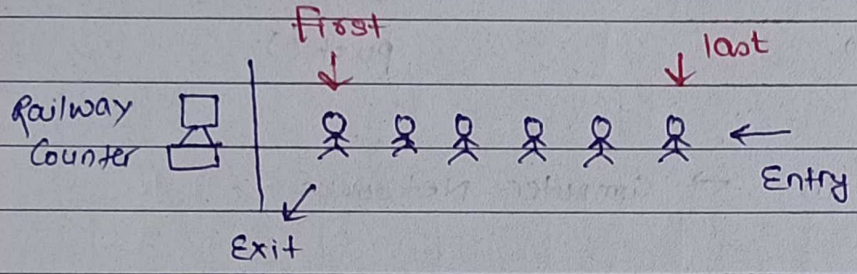
Lecture : 38

Queue : Basic

Queue → line (Any)

↓

Ex: Bank, Ticket Counter



Follow FIFO (First in first out)

Example:

You text you Friend on whatshapp:

Hii

Hello

How are you

I am Coming.

You send the 4 message:

→ First this message go into server and stored in a queue.

→ Then the message stored first in queue can go first to your friend whatsApp.

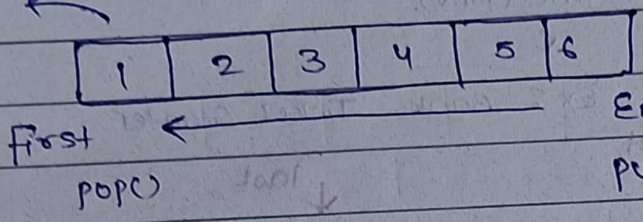
Teacher's Signature.....

* How to create a queue:

Queue creation

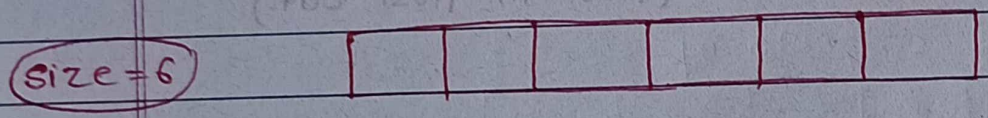
↳ By Array

↳ By Linked list



used in → Computer Networks

① Creation by array:



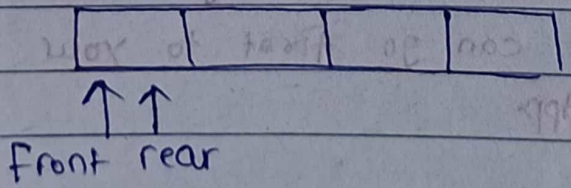
Two pointer

↳ front : from where we have to pop

↳ rear : from where we have push.

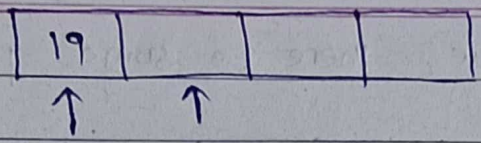
front always on first position.

When Queue is empty the front & rear both on first place.

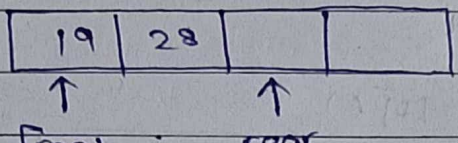


push(19)

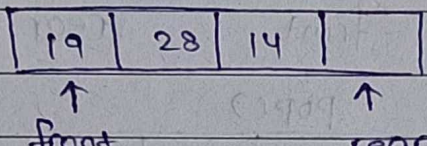
Teacher's Signature.....



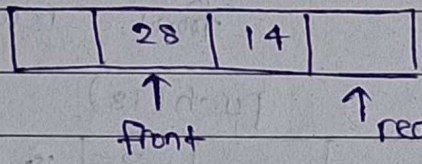
push(28)



Push (14)



pop()



code for push:

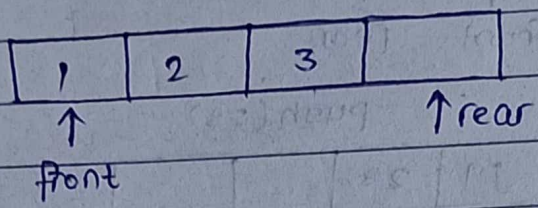
```
void push (int x) {
    if (rear == size) {
        cout << "Queue is full";
        return;
    }
    arr[rear] = x;
    rear++;
}
```

code for pop:

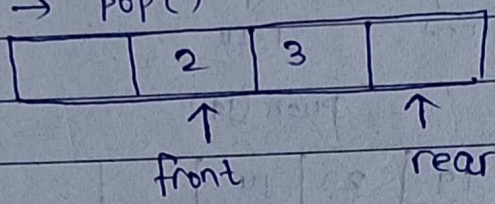
```
void pop() {
    if (front == rear) {
        front = rear = 0;
        cout << "Queue is empty";
        return;
    }
    front++;
}
```

Teacher's Signature.....

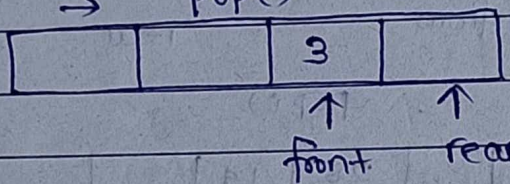
In this queue, there is some problem :-



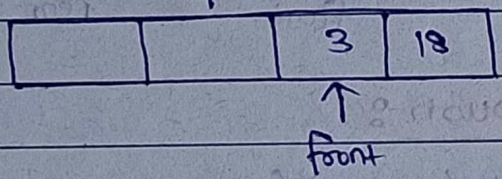
→ pop()



→ pop()



push(18)

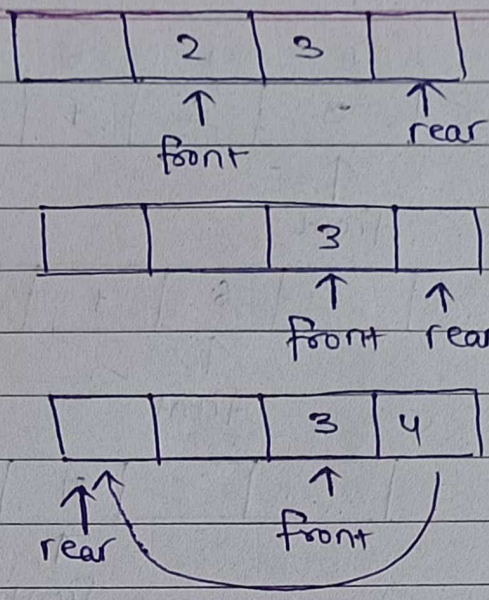


Now, if we want to add More element we can't able to add.

So, We make Circular Queue

अगर rear last तक आ जाएगा उसके बाद rear को First Block में भेज देंगे।

Ex:



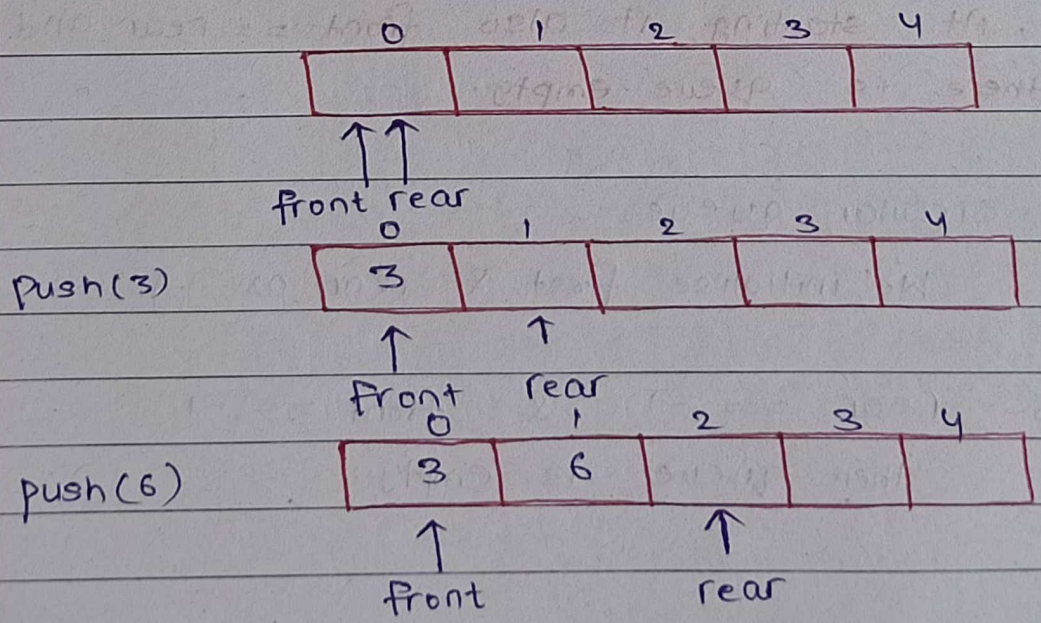
Condition :

```
rear = (rear + 1) %5;
```

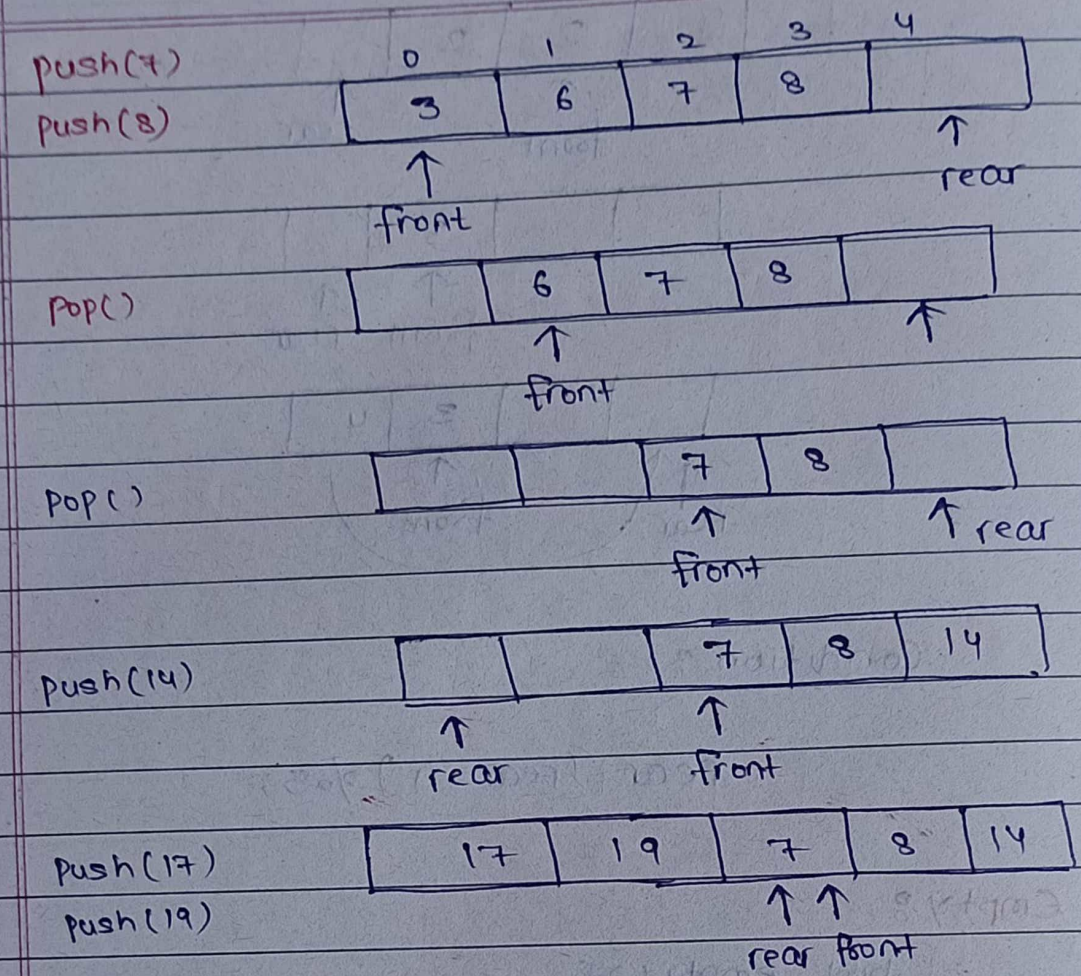
Empty :

```
bool empty(){
    return front == rear;
}
```

* Circular Queue : (How to implement)



Teacher's Signature.....



condition:

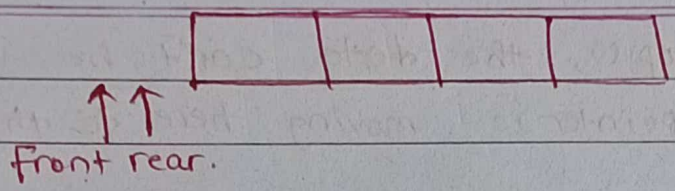
```
if (front == rear)
    queue full;
```

→ But, At starting, it also front == rear and there is queue empty.

In circular queue,

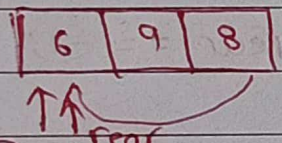
We initialise first & rear as -1.

if rear == -1 & front == -1
then queue is empty.

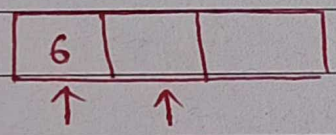


Code:

```
Void push(int x) {
    if ((empty())) {
        front = 0;
        rear = 0;
        arr[front] = x;
    }
```




```
    else if (full()) {
        Cout << "Queue is full";
    }
    else {
        rear = (rear + 1) % 5;
        arr[rear] = ~~data~~ x;
        return;
    }
```



Code: for pop():

```
Void pop() {
    if (empty()) {
        Cout << "Queue is Empty";
    }
    else if (front == rear) {
        Cout << "element popped" << arr[front];
        front = rear = -1;
        return;
    }
    else {
        Cout << "element popped" << arr[front];
        front = (front + 1) % size;
        return;
    }
}
```

Teacher's Signature.....

When We pop(), the data can't be deleted only front pointer is moving here to there.

Ex: Data recover:

Data is not deleted, it only move the pointer.

* Empty()

```
bool empty(){
    return front == -1 && rear == -1;
}
```

* full():

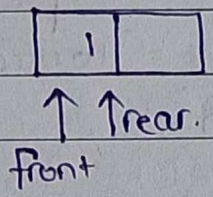
```
bool full(){
    return front = (rear+1) % size;
```

Teacher's Signature.....

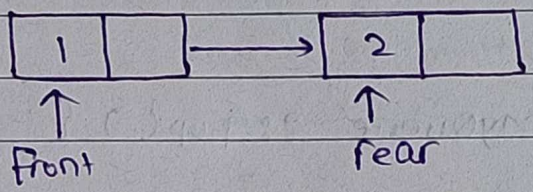
* Queue by LinkedList :

front = NULL;
rear = NULL; } → queue empty

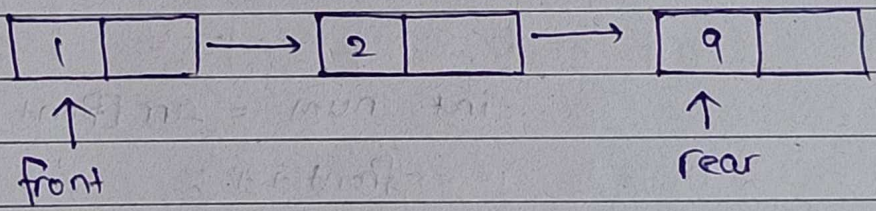
push(1)



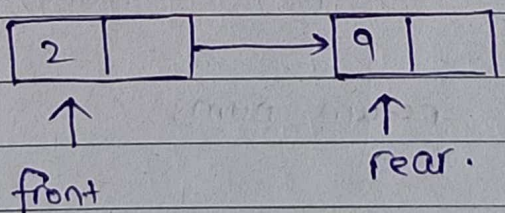
push(2)



push (9)



pop()



Page No.:
Date: / /

Problem 1: Implement Queue using array

Code:

```
Void MyQueue :: push(int x) {
                                  → size of array is mentioned in class.
    if (rear == 100005)
        return;

    arr[rear++] = x;
}

int MyQueue :: pop() {

    if (front == rear)
        return -1;

    int num = arr[front];
    front++;

    if (front == rear)
        front = rear = 0;

    return num;
}
```

Teacher's Signature.....

Page No.:
Date: / /

problem 2: Implement Queue using Linked list

```
Void MyQueue :: push (int x) {

    if (!front) {    // if Queue is empty.
        front = new QueueNode(x);
        rear = front;
        return;
    }
    else {
        rear → next = new QueueNode (x);
        rear = rear → next;
        ~~rear;~~
        return;
    }
}

int MyQueue : pop () {
    if (!front)
        return -1;

    else {
        int data = front → data;
        if (front == rear)
            front = rear = NULL;
        else
            Front = front → next;

        return data;
    }
}
```

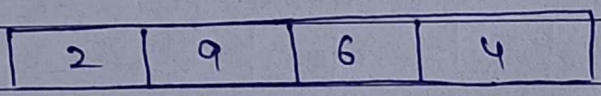
Teacher's Signature.....

Page No.
Date : / /

* Queue By STL:

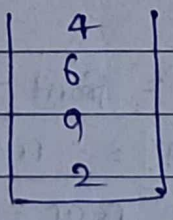
```
#include <Queue>
    int main() {
        queue<int> q;
            q.push(1);
            q.push(11);
            q.push(111);
            q.push(1111);
        cout << q.front() << endl;
            q.pop();
        return 0;
    }
```

* problem 3: Queue Reversal

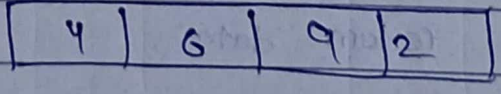


How to reverse this?

Store this in stack:-



Then one by one push into queue



You get the answer:-

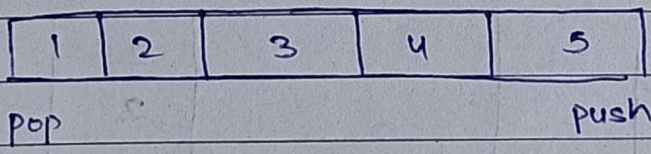
Teacher's Signature.....

Page No.:
Date: / /

code:

```
queue <int> rev (queue <int> q) {

    stack <int> s;
    while (q.size()) {  // copy from queue to
        s.push(q.front());      stack
        q.pop();
    }

    while( s.size()){      // copy from stack to
        q.push(s.top());       queue.
        s.pop();
    }
    return q;
}
```

* problem 4 : Queue using two stack :



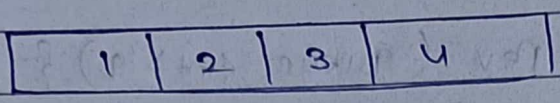
FIFO

How we can implement this queue using two stack?

Let's see.

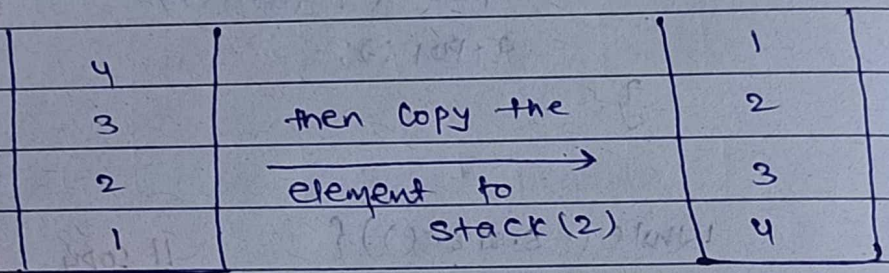
Teacher's Signature.....

For pop :



pop()
then 1 deleted

How we can do it from 2 stack.

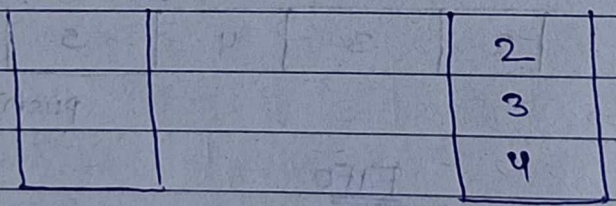


copy in stack 1.

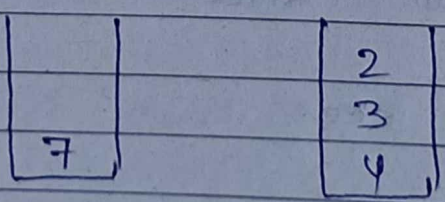
Now pop()
1 → deleted.

pop is performed perfectly using two stack.

For push :



Now we have to push 7.
push it in stack 1.



Teacher's Signature.....

If We Want to pop then pop it from stack 2.

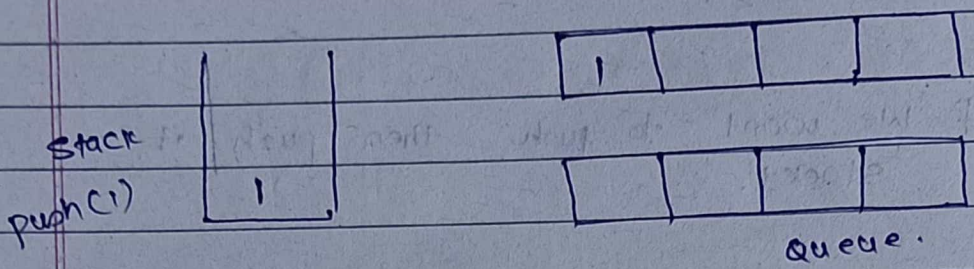
If We want to push then push it in stack1.

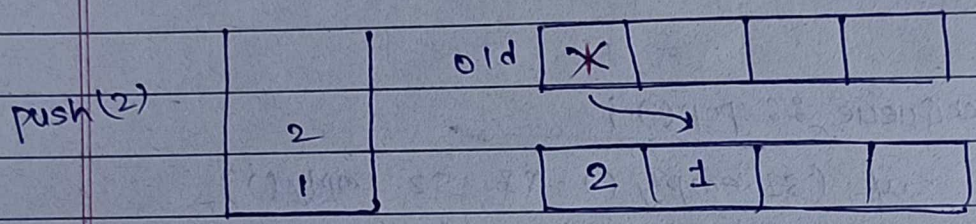
Code :

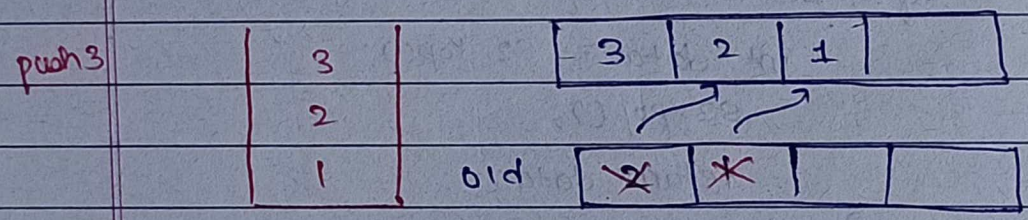
```
Void stackQueue :: pop() {
        if (s1.empty() && s2.empty())
            return -1;
        if (s2.size()) {
            int data = s2.top();
            s2.pop();
            return data;
        }
        while (s1.size()) {
            s2.push(s1.top());
            s1.pop();
        }
        int data = s2.top();
        s2.pop();
        return data;
}

Void stackQueue :: push(int x) {
        s1.push(x);
}
```

Teacher's Signature.....

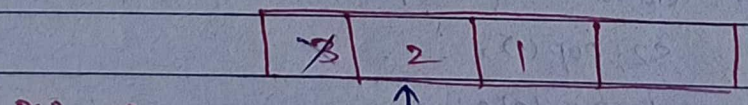
* problem - 5 stack using two Queues







Push always done is empty Queue and Copy all the element from another Queue.



Pop →

remove (3)

↳ first

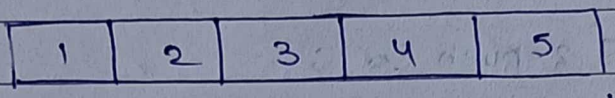
Code:

```
Void QueueStack :: push (int x) {

        q2.push (x);

        While (!q1.empty ()) {
            q2.push (q1.front ());
            q1.pop ();
        }

        swap (q1, q2);
}

int QueueStack :: pop () {

        if (q1.empty ())
            return -1;

        int topElement = q1.front ();
            q1.pop ();

        return topelement;
}
```

Teacher's Signature.....

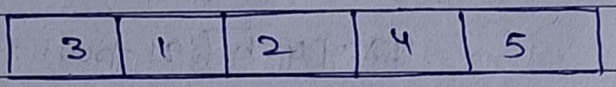
Problem : 6

Reverse First K elements of Queue :

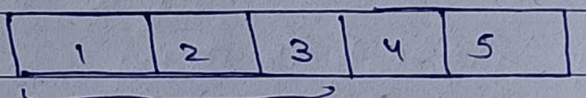


k = 3

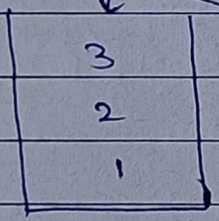
output :



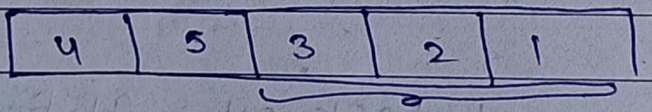
k = 3



k = 3

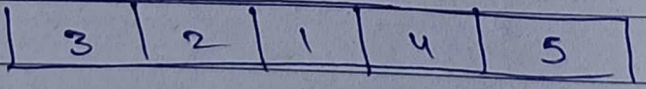


Now Copy in Queue



Remaining = 5 - 3 = 2

Pop & Paste back



output =

Page No.
Date : / /

Code :

```
queue <int> modifyQueue (queue <int> q, int k) {
    int a = k;
    stack <int> s;

    while (k) {
        s.push (q.front ());
        q.pop ();
        k--;
    }

    while (s.size ()) {
        q.push (s.top ());
        s.pop ();
    }

    int remaining = q.size () - a;

    for (int i = 0; i < remaining; i++) {
        int num = q.front ();
        q.pop ();
        q.push (num);
    }

    return q;
}
```

Teacher's Signature.....