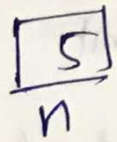
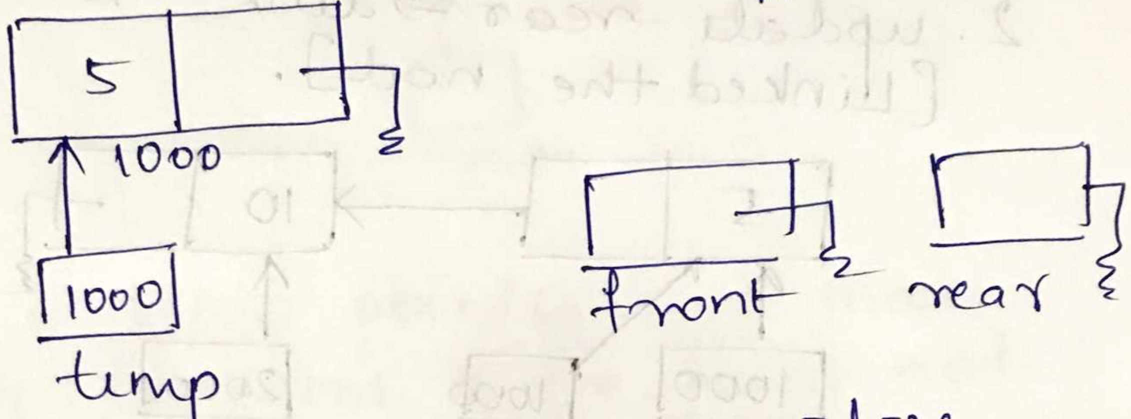
Linked List Implementation of Queues

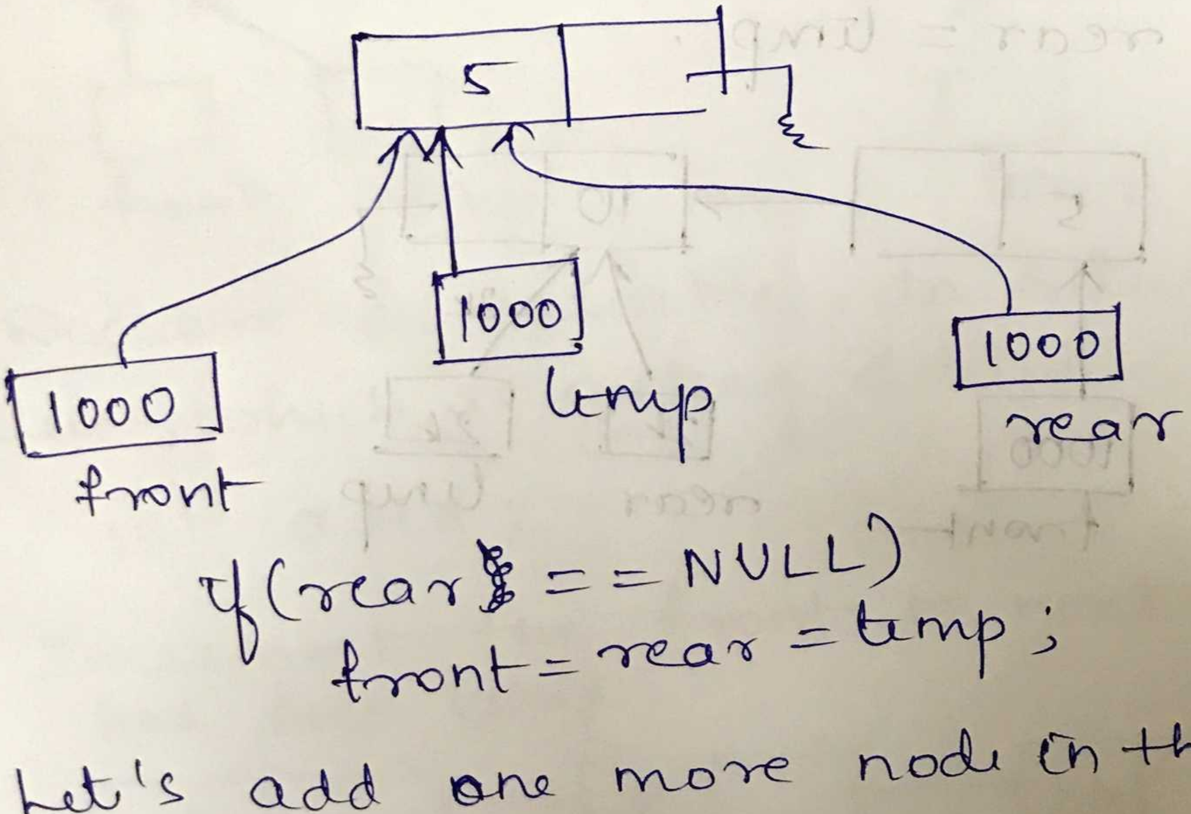
Enqueue

1. Create memory for node and initialize the node.



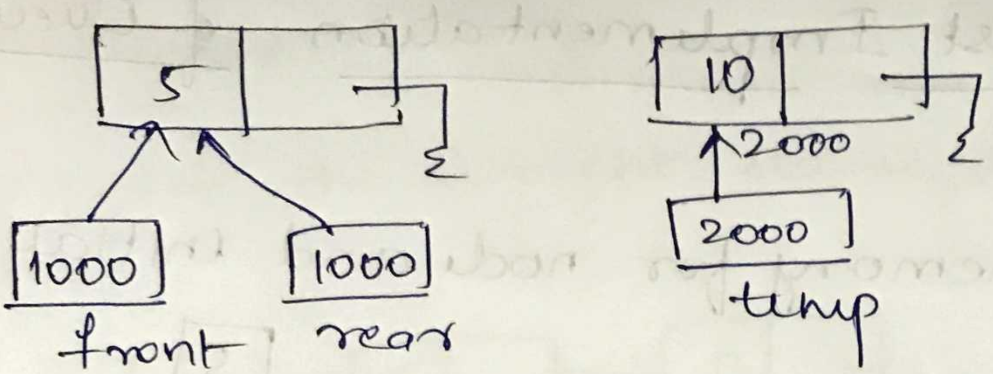


2. update front and rear pointers. Make them point to the temp.

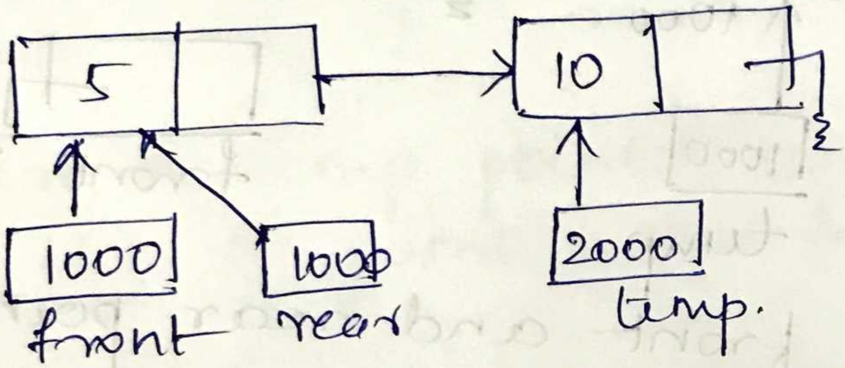


```
if(rear == NULL)
    front = rear = temp;
```

3. Let's add one more node in the queue.
   1. Create the node & Initialize.

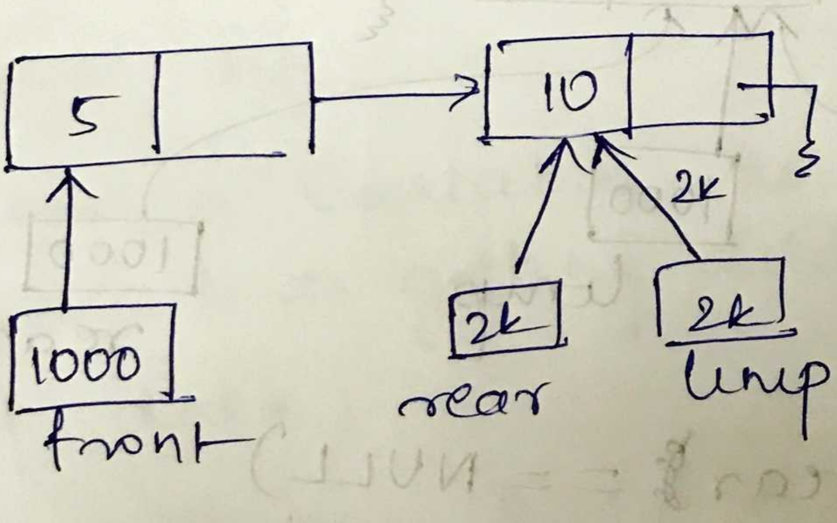


2. update rear → link to temp
[Linked the node].



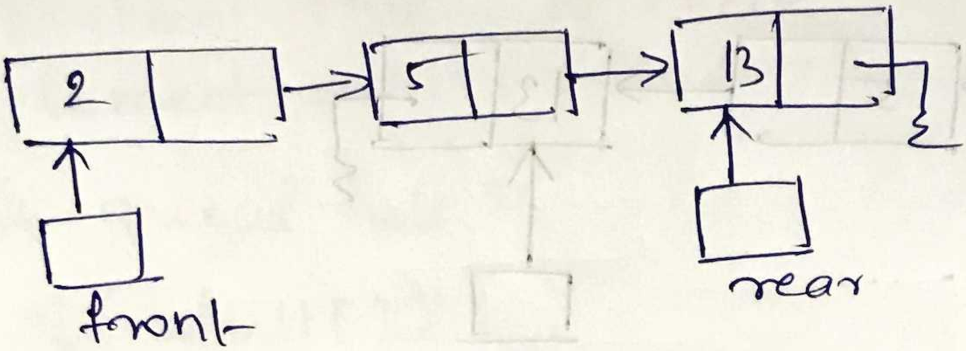
3. update ~~rear~~ rear to last node.

rear = temp.

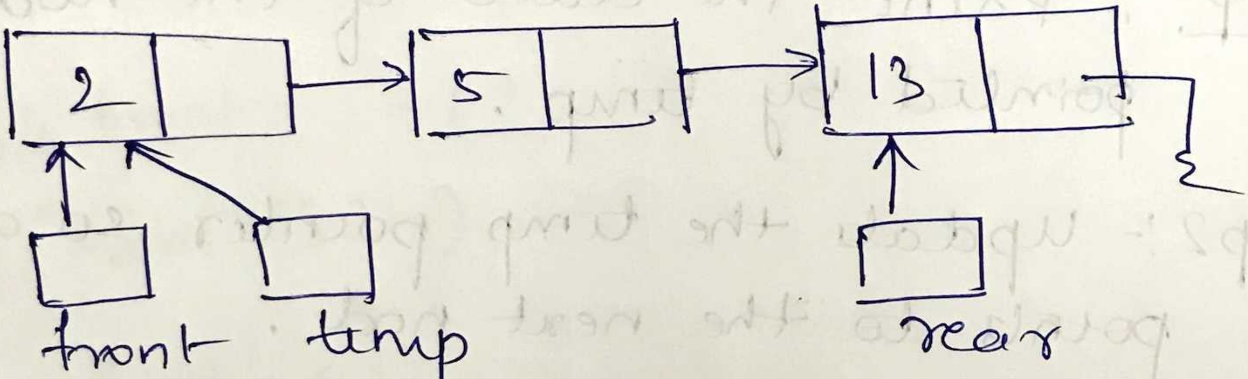


Dequeue

* Consider the Current status of the Queue.



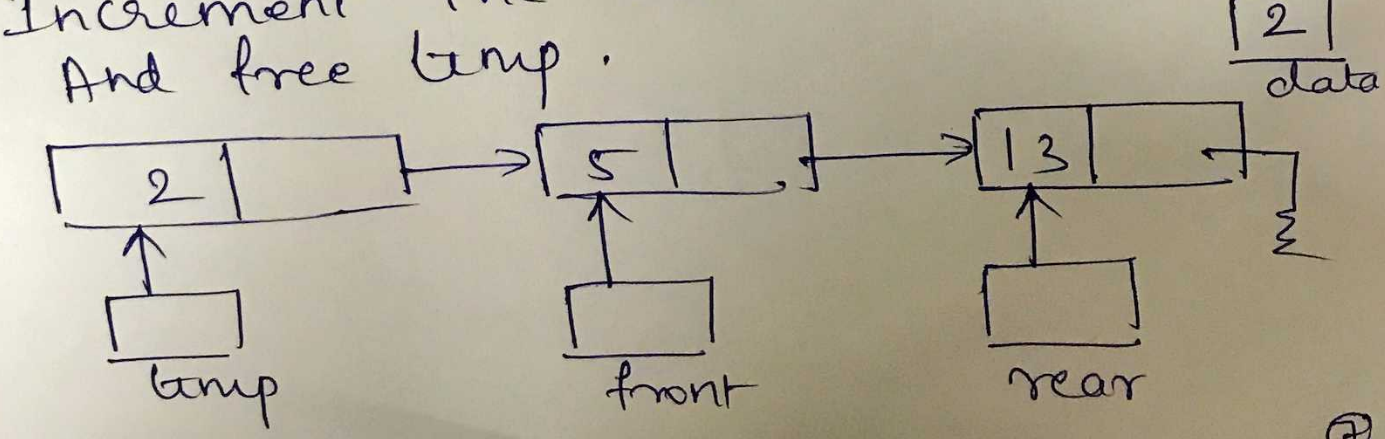
* Declare a temp pointer and make temp ptr to point to the front node.



* Declare a variable, to return the value, which is getting deleted.

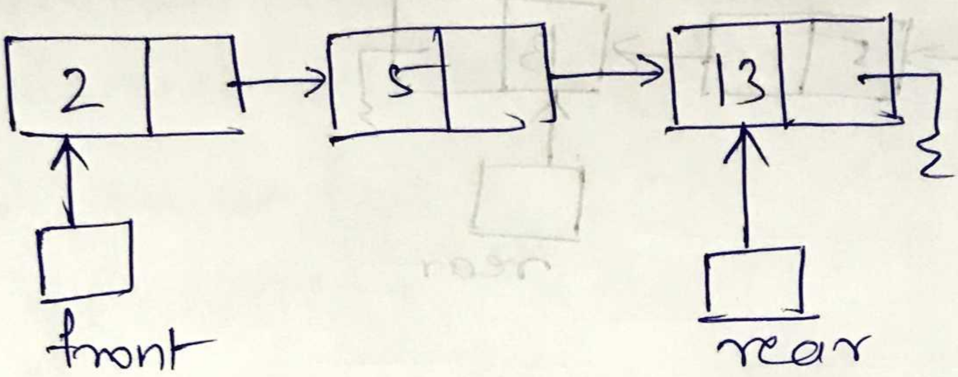
int data;

* Increment the front to next node. And free temp.



Print

* Consider the current status of the Queue.



* Declare a temp ptr to traverse.

Step1 :- Print the data of the node pointed by temp.

Step2 :- Update the temp pointer so as it points to the next node.

Step 3 :- Repeat step 1 and 2 until temp becomes NULL.

Implementation of circular Queue using array

Enqueue()

Important things to check before inserting an element in the circular queue:

* Is queue full ?

```
if(isfull())
{
    printf("Queue overflow \n");
    exit(1);
}
```

* ~~If~~ Is front == -1 ?

```
if (front == -1)
{
    front = 0;
```

* Is rear == MAX - 1 ?

```
if(rear == MAX-1)
{
    rear = 0;
}
```

dequeue

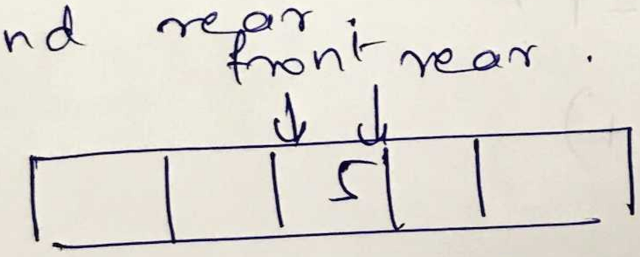
* Important thing to chuck before dequeuing.

* Is queue empty?

```
if(isEmpty())
{
    printf("Queue underflow\n");
    exit(1);
}
```

* Is queue has just one element? In this case we need to take care of front and rear.

Ex



-> 5 is only element

-> 5 is dequeued

-> then front and rear shld start from the beginning.

```
if (front == rear)
{
    front = -1;
    rear = -1;
}
```

*Is ~~queue~~. front == MAX-1 ?

```
if (front == MAX - 1) ?
if (front == MAX-1)
        front = 0 .
```

isfull( )

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

Ex1

5	4	3	9
0	1	2	3

for this Ex1
front==0 &&
rear==MAX-1 is
fine.

Ex2

		3	9
0	1	2	3

for Ex2 the above
condition
front ==0 &&
rear==MAX-1
does not hold
good, because this
is circular queue, ie the queue is not
empty.

Print()

*In case of circular queue, we have to consider the following scenarios:

* consider

1. front < rear

cqueue

25	10	5		
0	1	2	3	4

front = 0
rear = 2

2. front == rear

		5		
0	1	2	3	4

front = 2
rear = 2

3. front > rear

25	10		9	8
0	1	2	3	4

front = 3
rear = 1

In the above case, first we should print all the elements upto MAX-1 and then print all the elements from 0 to rear.