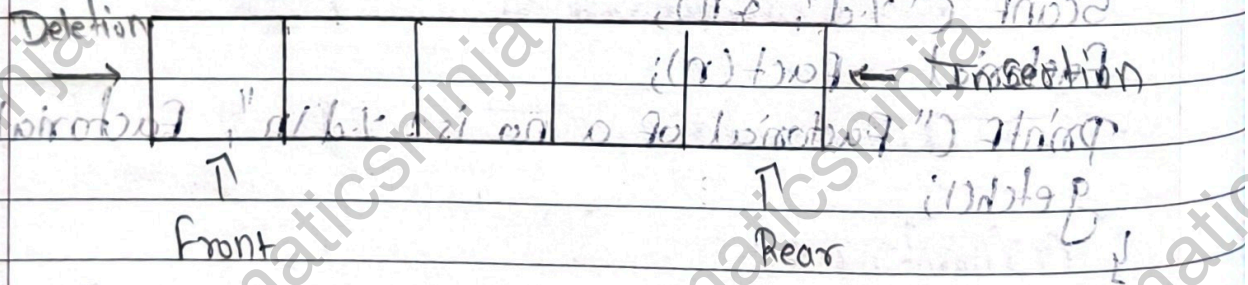
5. Queue

PAGE NO.:

Defination:
It is a special kind of list where atoms are inserted at one end (rear end) and deleted from the other end (front end).

- Queue is the FIFO List
- FIFo stands for First in First out.
That is the element insert First it will out First.



A queue datatype may be define formally as defined as follows:

```
typedef struct queue
{
    int data[max];
    int front, rear;
} queue;
```

An array representation of Queue require three entities
① An array to hold queue elements.
② A Variable to hold index of front element.

③ A variable to hold the index of rear element

During initialization of queue it's front and rear are set to -1.

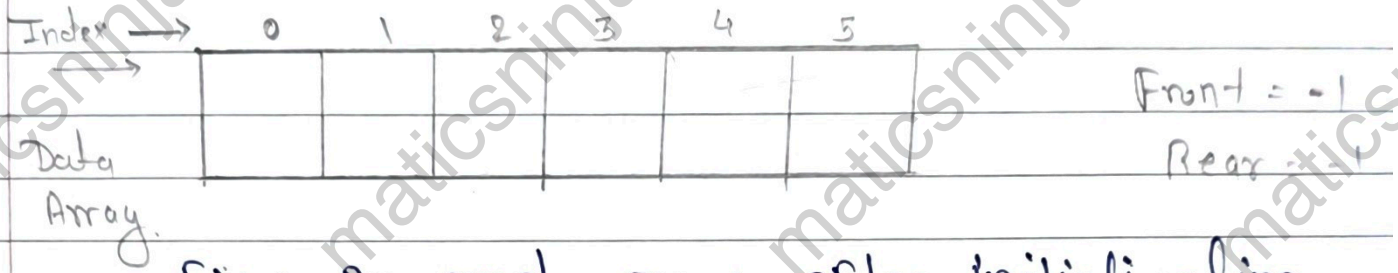


Fig: An empty queue after initialization.

- After insertion of element 4 queue will look like

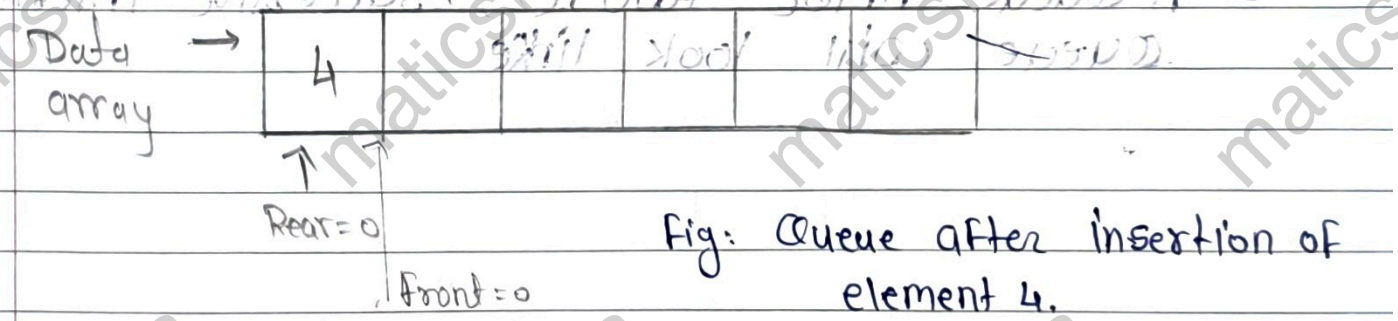


Fig: Queue after insertion of element 4.

- After insertion of 3 elements front remains at the same place where rear advances.

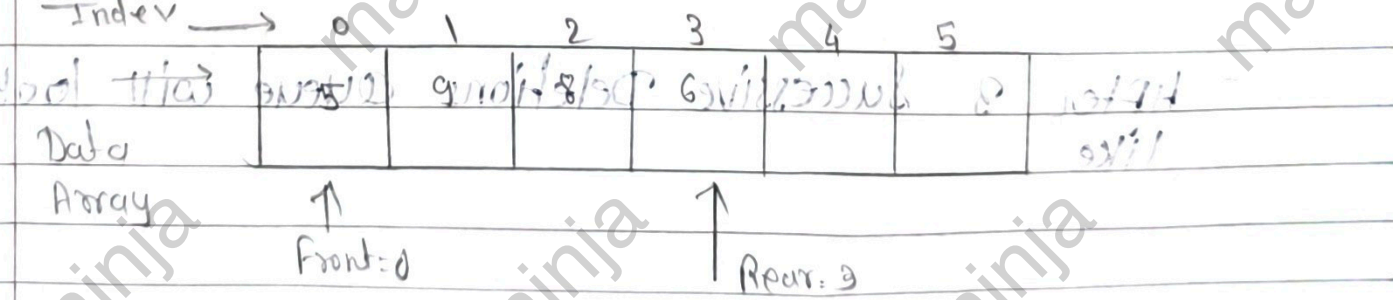


fig: queue after insertion of 3 elements is 9, 8, 6.

- A queue After Three Successive deletions queue will look like

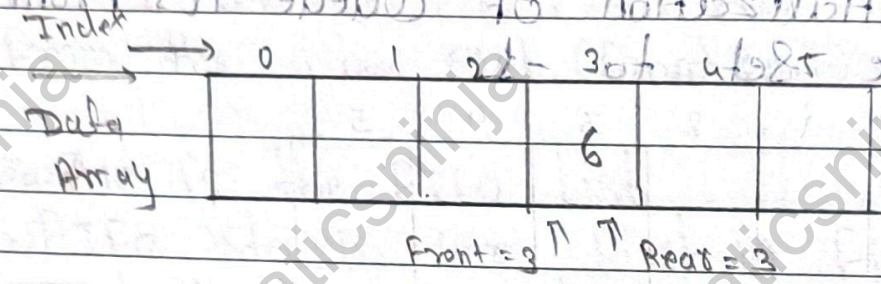


Fig: A queue After 3 Successive Deletion.

- A queue After Two (2) Successive insertion queue will look like

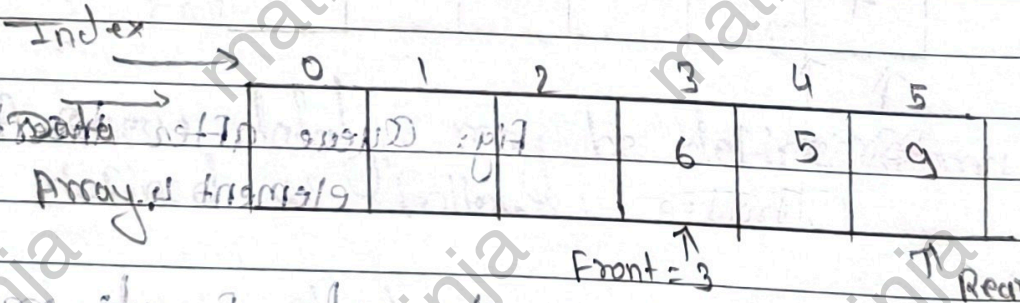


Fig: A queue After 2 Successive insertion

- After 3 Successive Deletions queue will look like

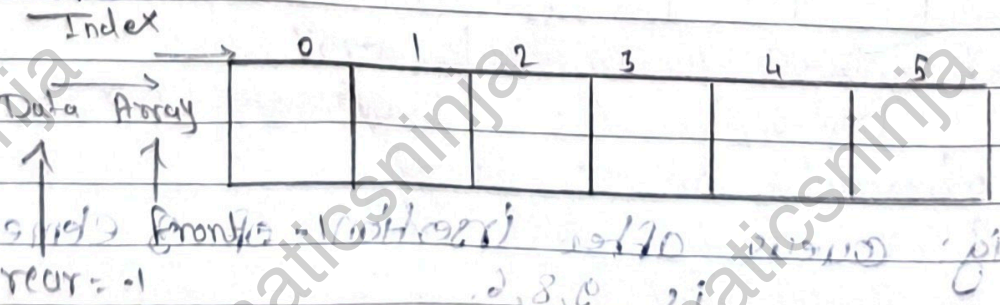


Fig: A queue After 3 Successive Deletions.

(i) if the queue is empty then
Front = 1 & rear = -1

(ii) if the queue is full
rear = Max-1

(iii) if rear = Front then queue contain just 1 element.

H/w

Q. Distinguish between Stack & Queue

- Queue overflow & Queue underflow.

Queue overflow
Is caused by insertion in a queue that is already full

Queue underflow
Deletion from an empty queue will caused a queue underflow

enqueue()
enqueue() opertion will cause overflow if the queue full.

dequeue()
dequeue() opertion will cause underflow if the queue is empty

operations on queue:

1] Initialise()
2] enqueue()
3] dequeue()
4] empty()
5] Full()
6] print()

1] C function to initialize:

```
Void initialize(Q *p)
{
p→rear = -1 ;
p→front = -1 ;
}
```

2] C function to check whether queue full or not.

```
int full(Q*p)
{
  if ( p→R = max-1)
   return (1);
  return(0);
  }
```

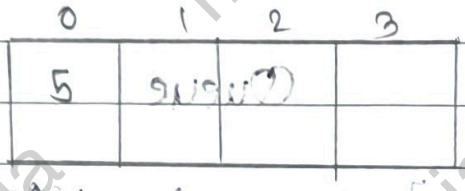
3] C function for empty()

```
int empty (Q *p)
  {
  if ( p→R == -1 )
  return (1);
  return(0);
  }
```

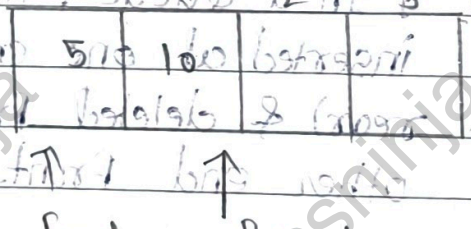
⇒ Distinguish between stack & queue:

Stack	Queue
1] Stack is a LIFO (last in first out) Structure.	1] Queue is a FIFO (First in First out) Structure.
2] In a stack, all insertion & deletion are permitted only at one end of the List	2] In a queue, items are inserted at one end (the rear) & deleted from the other end front.
3] Stack is always implemented using a linear array.	3] Queue should preferably implemented using a circular array.
4] Stack is widely used in recursion & processing of expression.	4] Queue is more useful in many of the real life applications.
5] It has only one pointer Variable	5] It has two pointer Variable
6] operations: 1. push() 2. pop()	6] operations: 1. enqueue() 2. dequeue()
7] No memory wastage	7] memory wastage in linear queue.

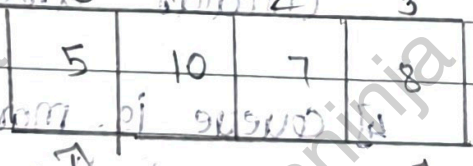
Insert 5 :-



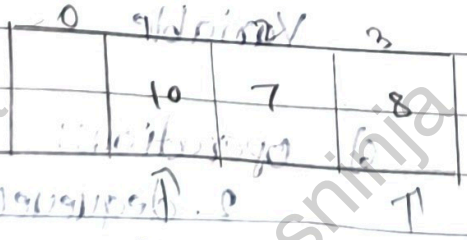
Insert 10 :-



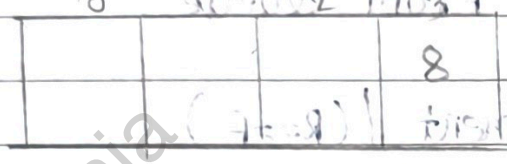
Insert 7 & 8



Delete



Delete 2 elements



Delete

front = -1
rear = -1

4] C function for enqueue()

Algorithm for insertion in a queue.

① Inserting in an empty queue
   a) R = F = 0
   b) data[R] = x

② Inserting in an nonempty queue.
   R = R+1
   data[R] = x

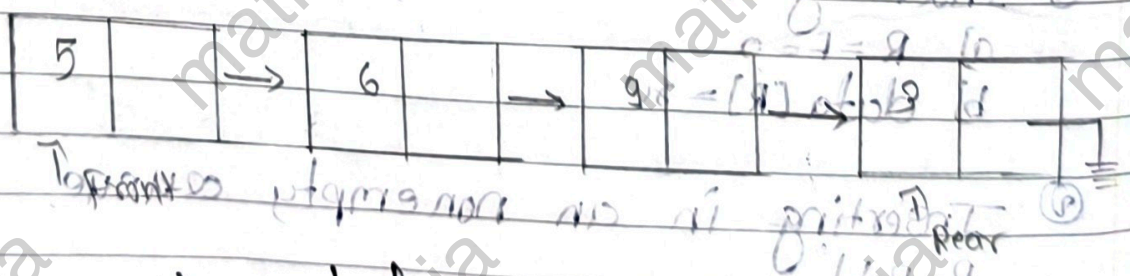
5] dequeue()
Algorithm for deletion from Queue

a] deletion of last element / (R==F)
① x = data[F]
② R = F = -1;
③ return (x);

b] deletion of element where Queue length >
① x = data[F]
② F = F+1
③ return (x);

⇒ Array & Link Representation of Queue.

Queue can be represented by Linked Structure of elements. Each element is stored in a node.



- Comparison between Array Representation & Link Representation of Queue.

- Maximum size of the queue is fixed at the time of compilation. When the queue is Represented using an array this causes wastage of storage space.

- In case of queue using linke structure no memory area is reserved in advanced.

- Memory for node acquire during runtime whenever insertion to be made.

- Array Representation of queue is simple to implement.

- In linked Representation Additional memory required to store address of next element.

- There is no such requirement in case of array Representation.

→ Types of queue.

1] Linear queue
2] Circular queue
3] Priority queue.

Advantages of circular queue:

Circular queue is linear data structure in which last node is connected back to the first node to form circle.

It follows FIFO principal

It is also known as Ring Buffer.

Element are inserted from rear. Deleted from front end.

There is potential problem with implement of queue using a simply array. The queue may apper to be full although there may be some space in the queue.

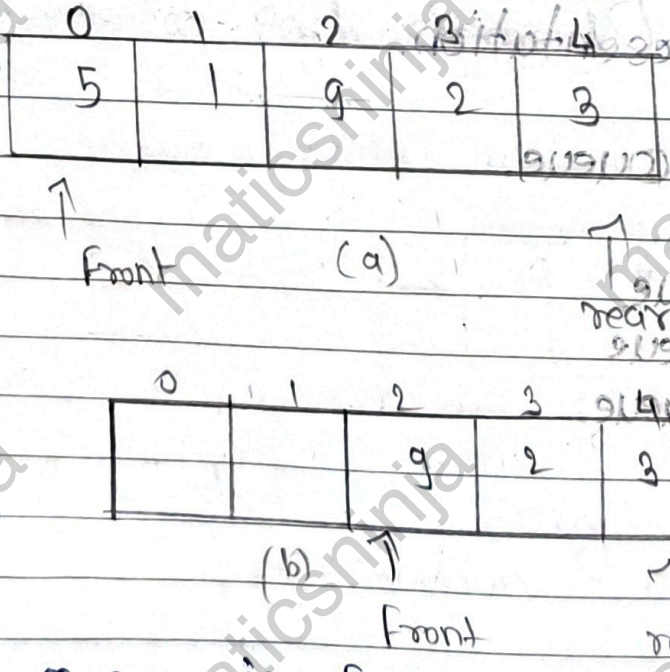


Fig: Queue is full although 0 & 1 locations are vacant.

- After insertion of 5 elements in the array is rear = 4 front = 0

- After two successive deletion front = 2 and rear = 4

- Queue in the Fig b is Full no empty space head of the rear.

- So To overcome this problem Simple Solution is whenever rear gets to the end of array it is wrapped around to the beginning now the array can be thought as a circle

- The first position follows the last element.

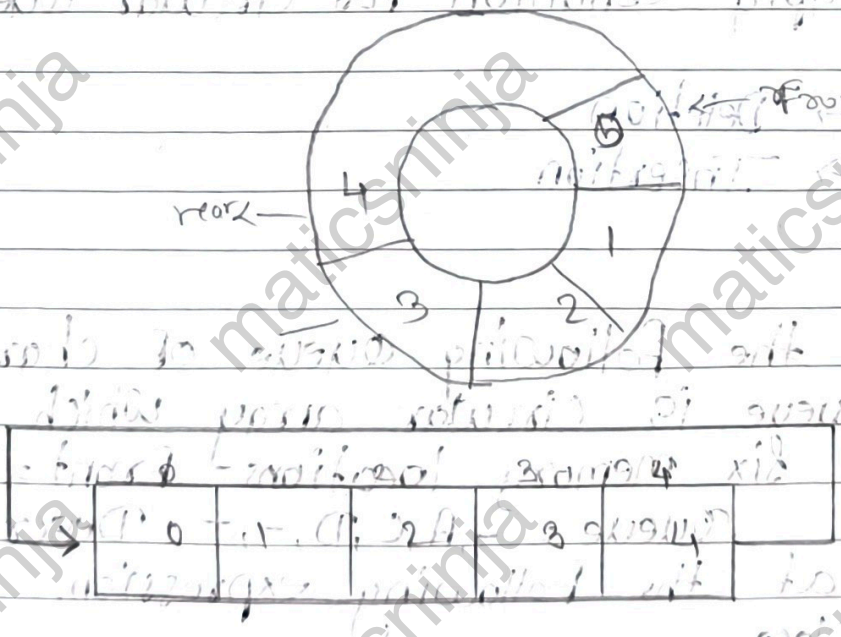


Fig. Circular array.

The condition for Circular queue is Full.

Front = (rear + 1) % max

0	1	2	3	4
5	4	9	2	8

Front ↑ Rear ↑

Front = (rear + 1) % max
= 4 + 1 % 5
= 5 % 5
= 0

Q. Explain with suitable diagram queue full and queue empty condition for circular queue.

Front → Deletion
Rear → Insertion

Q Consider the following queue of characters where queue is circular array which is allocated six memory locations Front = 2 rear = 4 Queue = -, A, C, D, -, - Describe queue at the following expression. take place.

	operation	queue	Rear	front
1]	Initially	–, A, C, D, F, –	4	2
2]	F is added	–, A, C, D, F, –	5	2
3]	Two letters are deleted	–, –, –, D, F, –	5	4
4]	K, L, M are added	L, M, –, D, F, K	2	4
5]	Two letters are deleted R is	L, M, –, –, –, K	2	6
6]	R is added	L, M, R, –, –, K	3	6
7]	Two letters are deleted.	–, M, R, –, –, K	3	1
8]	one letter is deleted	–, –, R, –, –, K	3	3
9]	5 is added	–, –, R, 5, –, K	4	3

3] Priority queue

Priority queue is an orderlist of Homogeneous elements. In the normal queue, service is provide on the basics of FIFO.

In a priority queue service is not provid on the basics of FCFS, rather than each element has a priority based on the urgency of need.

An element with higher priority is process before all elements with lower priority

Elements with same priority are processe on FCFS.

Ex:
Hospital waiting queue / Room

Other Application of priority is found in long term scheduling.

- Application of queue.

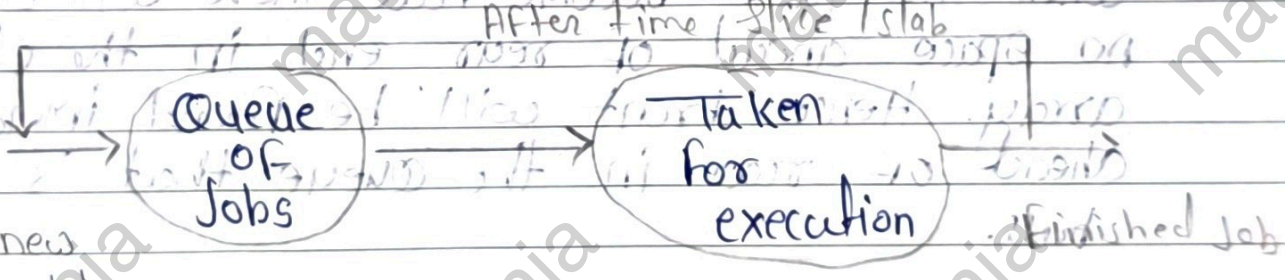
1] Job / cpu process scheduling
2] Spooling
3] Queue simulation

Queue is very useful datastructure. Various fearsturs of OS (operating system) are implemented using queue.

1. Scheduling process (round robbin)
2. Spooling
3. Queue simulation

1] Job scheduling
Jobs enters in the systemes are stored in queue. The jobs can be selected, for execution as per scheduling algorithm.

round robbin technique.



Suppose there are n jobs waiting for execution.
In round robbin technique cpu gives fixed quantum / slice of time to each job

① A job is selected from job queue.
② It is executed for fixed slice of time.
③ If the job is not completed by this time then it is removed & pushed back in queue of jobs.

Queue Simulation

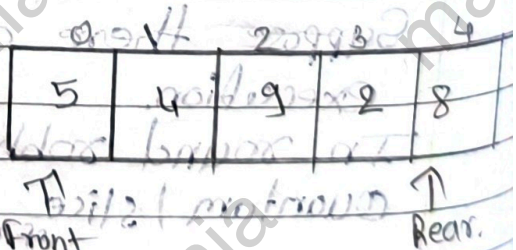
Queue Simulation involvs
1] arrival time
2] Service time
3] Spooling

It is used to maintain queue of job to be printed.

Q. Explain with suitable diagram queue full and queue empty condition for circular queue

→ Queue Full:
When the queue become full there will be no space ahead of rear end in the circular array. Hence front will be found immediate ahead of rear in the queue that is full.

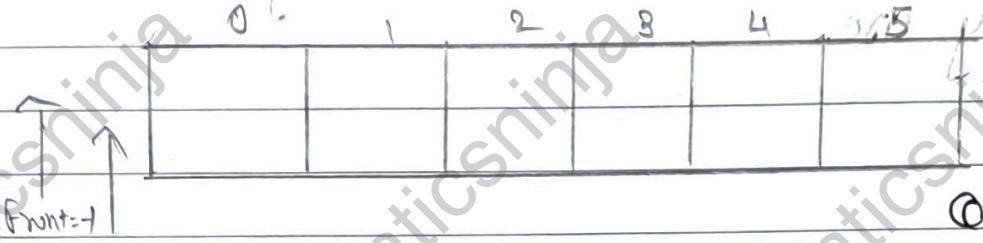
The queue full condition
front == (rear+1) % max
0 == (4+1) % 5
0 == 5 % 5
0 == 0
=



Queue Empty
When there is no element found in the queue then queue become empty.

The queue empty condition

front = -1 & rear = -1



Queue Empty

When the queue is empty both rear & front will be equal to -1.