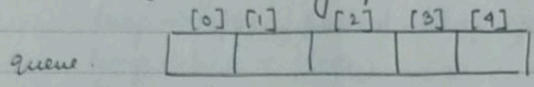
## Queue :

Queue is an ordered list of elements in which we can add elements only at one end, called the rear of the queue and delete elements only at the other end called the front of the queue. Its behaviour seems to be first in first out. So, queue is called FIFO data structure.

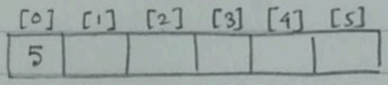
eg. in daily life we see that queue of people.



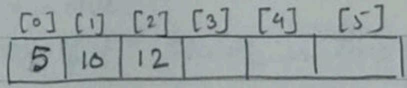
(a) Empty Queue.

front = -1
rear = -1

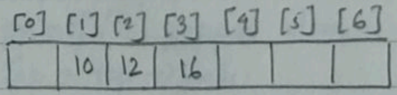
(if there is no element)



front = 0
rear = 0.



front = 0
rear = 2



front = 1
rear = 3

### Implementation of Queue :

Queue can be represented in two ways –

i) Array implementation of queue.
ii) Linked list implementation of queue.

i) Array implementation of Queue.

In queue, we add element at the rear end and delete the element at the front end.
So, we take two variables rear and front in array implementation of queue.
There may be two possibility cases.

i) when there is no place for adding elements in queue. then this is called overflow condition.
So first we check the value of rear with size of array.

ii) The second possibility arises when there is no element for deleting from queue, then value of front and rear will be -1 or front will be greater than rear. So we check the condition before deleting the element of queue.

ADD operation in queue :

```
if (rear == MAXSIZE - 1)
        printf (" Queue Overflow\n");
else
    {   if (front == -1)
            front = 0 ;
            rear = rear+1;
            queue [rear] = added_item ;

    }
```

Here initially front and rear are -1.
After adding first element both will become 0 and after that only rear will increase when we add element in queue.

DATE

DELETE operation in queue :

```
if (front = = -1 || front > rear)
   {    printf (" Queue Underflow\n");
        return;
   }
else
   {
        printf ("Element deleted from queue is : %d \n",
                     queue [front]);
        front = front + 1;
   }
```

Here underflow condition can arise in both situations if front is -1 or front is greater than rear.
For deleting the element, we increase the front value by 1.

—x—

Nat

ii) Linked List Implementation of Queue

The structure of a node -

```
struct node {
        int data;
        struct node * link;
    }
```

Add operation in queue :

For adding the element in queue we add the element at the end of linked list. Here front will point to the first node of linked list and rear will point to the last node of linked list.

```
temp -> info = added_item;
temp -> link = NULL;
if (front = = NULL)
        front = temp;
else
        rear -> link = temp;
rear = temp;
```

Delete Operation in Queue :

For deleting the element of a queue, we delete the first node of the linked list. Since queue is a first in first out structure and first node of the linked list will be the first element of the queue.

```
if (front = = NULL)
        printf (" Queue Underflow \n");
else
    {   temp = front;
        printf (" Deleted element is %d \n", temp -> info);
        front = front -> link;
    }   free (temp);
```

Here front is pointing to the first element of queue.
After deleting the element it will point to the next element of queue

Q.8 Write a program of queue using array.

```
#include <stdio.h>
#define MAX5
int queue [MAX];
int rear = -1;
int front = -1;
main ()
{
    int choice ;
    while(choice != 4)
    {
        printf ("\n. Insert");
        printf ("\n. Delete");
        printf ("\n. Display");
        Printf ("\n. Quit");
        printf ("\n. Enter your choice: ");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1 :  insert ();
                      break;
            case 2 :  delete ();
                      break;
            case 3 :  display ();
                      break;
            case 4 :  exit (1);
            default : printf ("\n Wrong choice");
        }
    }
}
```

```
insert ( )
{
        int added_item ;
        if ( rear = = MAX -1 )
                printf ( " Queue Overflow \n " );
        else
        {       if ( front = = -1 )
                        front = 0 ;
                printf ( " Input the element for adding in queue:" );
                scanf ( "%d", & added_item );
                rear = rear + 1 ;
                queue [rear] = added_item ;
        }

del ( )
{
        if ( front = = -1 || front > rear )
            {   printf ( " Queue Underflow \n" ) ;
                return ;
            }
        else
        {   printf ( " Element deleted from queue is %d \n",
                        queue [front] ) ;
            front = front + 1 ;
        }
}
```

```
display()
{
        int i;
        if (front == -1)
            printf ("Queue is empty\n");
        else
        {    printf ("Queue is : \n");
             for(i=0; i<=rear ;i++)
                    printf ("%d", queue [i]);
             printf("\n");
        }
}
```