QUEUE

Queue is a linear data structure in which elements (data) are inserted from one end called **rear** (also called tail) and elements are deleted / removed from other end call **front** (also called head).

Queue follows **FIFO** (first in first out) order / basis, which means that element inserted first will be removed first. Which is exactly how queue system works in real world.

Note: Queue follows LILO (Last in Last Out) order also.

Examples) A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world examples can be seen as queues at the ticket windows and bus-stops.







Basic Operations on Queue:

- Enqueue: Add an element to the end of the queue
- Dequeue: Remove an element from the front of the queue
- IsEmpty: Check if the queue is empty
- IsFull: Check if the queue is full
- Peek: Get the value / element of the front of the queue without removing it

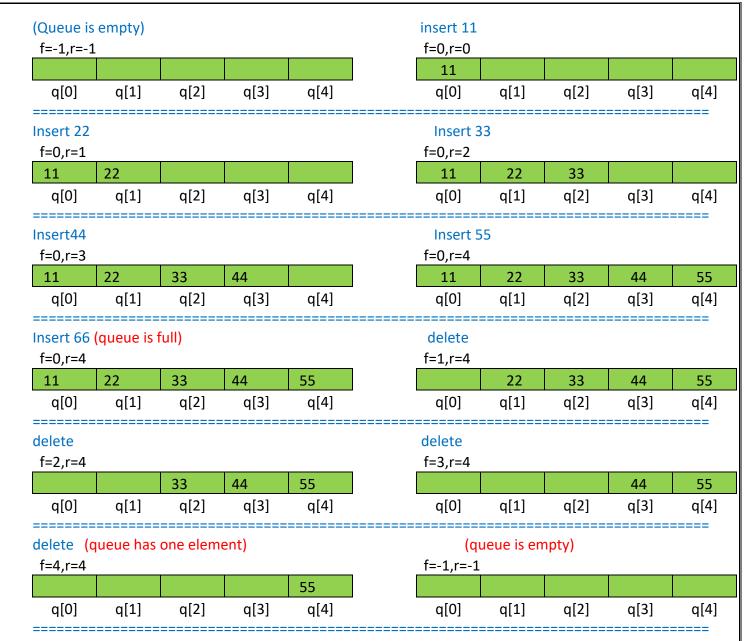
Queue Implementation:

Queue can be implemented in two ways

1) Array method 2) Linked list method

The below diagrams represent enqueue & dequeue operations on queue using **Array method** and the position of front and rear.

Example 1) int q[5], f=-1, r-1;



Note:

- For each enqueue / insert operation rear is incremented (by one)
- For each dequeue / delete operation front is decremented (by one)

Applications of Queue:

- 1. Operating systems schedule jobs (with equal priority) in the order of arrival (e.g., a print queue).
- 2. Serving requests on a single shared resource, like a printer, CPU task scheduling etc.
- 3. In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free.

- 4. Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive i.e First come first served.
- 5. When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

// Write a C program to implement queue and its operations using the arrays.

```
#include<stdio.h>
#include<stdlib.h>
#define QSIZE 5
int f=-1,r=-1; // global variable
void enqueue(int q[], int item)
     if(r==QSIZE-1) // if(-1==4)
        printf("queue is full \n");
     else
      {
        if((f==-1)\&\&(r==-1))
          f=r=0;
       else
         r++;
      q[r]=item;
     }
return;
}
void dequeue(int q[])
{
    int item;
    if((f==-1)\&\&(r==-1))
      printf("queue is empty \n");
    else
     {
        item=q[f];
        printf("the deleted item from queue is %d\n",item);
        if(f==r) // if queue contains only one single item
           f=r=-1; // make queue as empty
        else
           f++;
 return;
```

```
void display(int q[])
       int i;
       if((f==-1)&&(r==-1))
         printf("queue is empty\n");
        else
        printf("\n elements in queue ");
        for(i=f;i<=r;i++)
           printf("%d \t ",q[i]);
        }
return;
}
void main()
  int q[QSIZE], item, choice;
  do{
       printf(" \n 1. insert \n 2. delete \n 3. display \n 4. exit\n");
       printf("enter your choice \n");
        scanf("%d",&choice);
       switch(choice)
       case 1: printf("enter price of the item \n");
               scanf("%d",&item);
               enqueue(q,item); break;
       case 2: dequeue(q);break;
        case 3: display(q);break;
        case 4: exit(0); // break;
        default : printf(" invalid choice \n");
}while(choice!=4);
```