KJ's Educational Institutes
K J College Of Engineering & Management Research, Pune.
Department of E & TC

CLASS: S. E. (E &TC) **SUBJECT:-DSA**

Ex. No: 2 **Date:**

AIM: Stack or Queue using Array (Static Implementation)

Simulate a parcel handling system at a post office where packages are stacked (LIFO) or queued (FIFO).

Use an array to implement a stack (push, pop, display) or a queue (add, delete, display). Choose the appropriate model based on the scenario.

OBJECTIVES

This assignment will help the student to realize the implementation difference between stack and queue. Also this will clear the implementation concepts queue.

THEORY

The **queue** data structure is characterized by the fact that additions are made end, or *tail*, of the queue while removals are made from the front, or *head*, of the queue. For this reason, a Queue is referred to as a FIFO structure (First-In First-Out). Queues occur naturally in situations where the rate at which clients' demand for services can exceed the rate at which these services can be supplied. For example, in network where many computers share only a few printers, the print jobs may accumulate in print queue. In an operating system with a GUI, applications and window communicate using messages, which are placed in message queues until they can be handled.

Operations:

The main primitive operations of a queue are known as:
Add : adds a new node
Remove: removes a node

Additional primitives can be defined:

IsEmpty: reports whether the queue is empty
IsFull: reports whether the queue is full
Initialize: creates/initializes the queue
Destroy: deletes the contents of the queue (may be implemented by re-initializing the queue)

Static implementation of queue:

The implementation of queue using sequential representation is done by using one dimensional array called 'q' of some size MAX & two integer variable front and rear initially front and rear is set to -1 whenever new element is to be added it is added from rear & when element is removed it is removed from front.

```
Struct queue
    {
     Int q[MAX];
     Int front,rear;
     }
```

Dynamic implementation of queue:

Linked representation is used for dynamic implementation of queue,
In linked representation there is no memory constraint. size can be increased at run time.
We can also release the memory when the element is removed from queue.
To create queue using Linked representation following steps are implemented:

1) Front and rear are NODE type pointer, which are NULL.

2) Whenever new node is created set rear to new.

3) Whenever node is to be deleted store address of front into temp and decrease front and release the temp.

```
Struct node
{  int data;
Struct nide * link;
}
```

Input :

Input is the data/element that is to be added to the queue.

Output:

Output is the data/element removed from the queue, data elements in the queue.

Application :

- Job scheduling in compiler.
- Queue simulation.

CONCLUSION:-