Practical No -11

Title: Implement queue for processing job in operating system.

Aim: Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.

Prerequisite:

- Basics understanding of Queue

Objectives:

- To understand implementation of Queue

Input: Sequence of job.

Outcome:

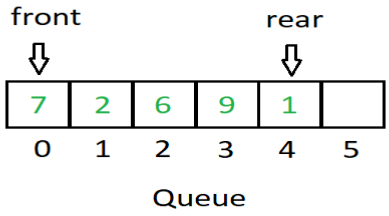
- At end of this experiment, student will be able to illustrate the concept of queue in data structure.

Theory:

Queue is an abstract data structure, somewhat similar to Stack. In contrast to Queue, queue is opened at both end. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

Queue Representation

As we now understand that in queue, we access both ends for different reasons, a diagram given below tries to explain queue representation as data structure



Basic Operations
Queue operations may involve initializing or defining the queue, utilizing it and then completely erasing it from memory. Following are basic operations associated with queues –

enqueue() - add (store) an item to the queue.

dequeue() - remove (access) an item from the queue.

Few more functions are required to make above mentioned queue operation efficient. These are -
isfull() - checks if queue is full.

isempty() - checks if queue is empty.

In queue, we always **delete** (or access) data, pointed by **front** pointer and while inserting (or storing) data in queue we take help of **rear** pointer.

Conclusion:

Thus we have implemented C++ program for the job queue for the operating system.