**Practical No: 11(E)**

**Practical Title:** Perform different operations on Queue

**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.

**Pre-requisite:**

* Basics of Queue
* Different operations that can be performed on queue

**Objective:** To perform addition and deletion operations on queue.

**Input :**

* Size of queue
* Elements in queue

**Outcome:**

* + Result of addition of job operation on queue.
  + Result of deletion of job operation on queue.

**Theory:**

Write theory of queue (definition, concepts, types, advantages, disadvantages)

Explain queue as an ADT. (write pseudo code)

**Algorithms :**

Step 1: Include all the header files which are used in the program and define a constant 'SIZE' with specific value.

Step 2: Declare all the user defined functions which are used in queue implementation.

Step 3: Create a one dimensional array with above defined SIZE (int queue[SIZE])

Step 4: Define two integer variables 'front' and 'rear' and initialize both with '-1'. (int front = -1, rear

= -1)

Step 5: Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

enQueue(value) - Inserting value into the queue:

In a queue data structure, enQueue() is a function used to insert a new element into the queue. In a queue, the new element is always inserted at rear position. The enQueue() function takes one integer value as parameter and inserts that value into the queue. We can use the following steps to insert an element into the queue...

Step 1: Check whether queue is FULL. (rear == SIZE-1)

Step 2: If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.

Step 3: If it is NOT FULL, then increment rear value by one (rear++) and set queue[rear] = value.

deQueue() - Deleting a value from the Queue:

In a queue data structure, deQueue() is a function used to delete an element from the queue. In a queue, the element is always deleted from front position. The deQueue() function does not take any value as parameter. We can use the following steps to delete an element from the queue...

Step 1: Check whether queue is EMPTY. (front == rear)

Step 2: If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.

Step 3: If it is NOT EMPTY, then increment the front value by one (front ++). Then display queue[front] as deleted element. Then check whether both front and rear are equal (front == rear), if it TRUE, then set both front and rear to '-1' (front = rear = -1).

display() - Displays the elements of a Queue:

We can use the following steps to display the elements of a queue...

Step 1: Check whether queue is EMPTY. (front == rear)

Step 2: If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.

Step 3: If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front+1'.

Step 3: Display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i' value is equal to rear (i <= rear)

Flowchart :

Draw flowchart for above algorithms.

**Conclusion:** By this way, we can perform different operations on queue

**Question Bank:**

1.What is Queue?

2.What are the different operations that can be performed on queue?

3.Explain all the operations on queue

4.Which are different types of queues AND Explain.