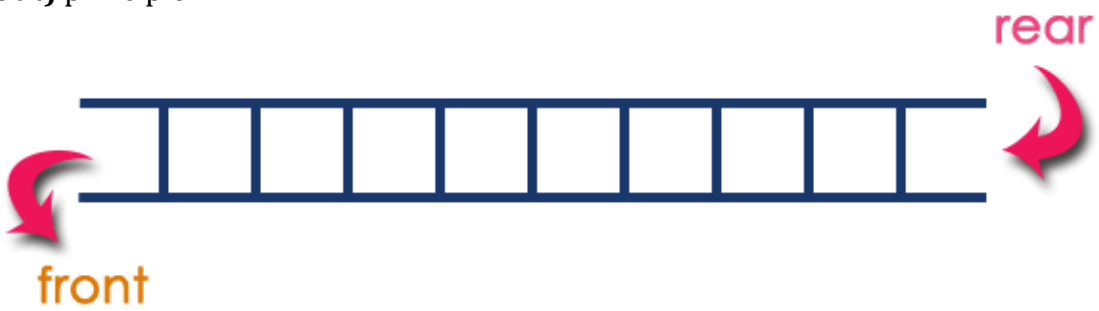
Practical # 08

Objective:

Demonstrate *Queue and discuss all the operations performed on Queue.*

Theory:

In this Lab, we discuss the Queue data type. Queue is a linear data structure in which the insertion and deletion operations are performed at two different ends. In a queue data structure, adding and removing elements are performed at two different positions. The insertion is performed at one end and deletion is performed at another end. In a queue data structure, the insertion operation is performed at a position which is known as '**rear**' and the deletion operation is performed at a position which is known as '**front**'. In queue data structure, the insertion and deletion operations are performed based on **FIFO (First In First Out)** principle.



Basic Operations

The following operations are performed on a queue data structure:

1. **enQueue(value) - (To insert an element into the queue)**
2. **deQueue() - (To delete an element from the queue)**
3. **display() - (To display the elements of the queue)**

Queue data structure can be implemented in two ways. They are as follows

1. **Using Array**
2. **Using Linked List**

When a queue is implemented using an array, that queue can organize an only limited number of elements. When a queue is implemented using a linked list, that queue can organize an unlimited number of elements.

Queue Operations using Array

Queue data structure using array can be implemented as follows: Before we implement actual operations, first follow the below steps to create an empty queue.

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 a 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 4 - Display '**queue[i]**' value and increment '**i**' value by one (**i++**). Repeat the same until '**i**' value reaches to **rear** (**i** <= **rear**)

Lab Objectives:

- To be able to perform fundamental operations on Queue.

C++ program: Write C++ program to implement QUEUE using Array.

```
#include <iostream>
using namespace std;
int queue[100], n = 100, front = - 1, rear = - 1;
void Insert() {
   int val;
   if (rear == n - 1)
   cout<<"Queue Overflow"<<endl;
   else {
      if (front == - 1)
      front = 0;
      cout<<"Insert the element in queue : "<<endl;
      cin>>val;
      rear++;
      queue[rear] = val;
   }
}
void Delete() {
   if (front == - 1 || front > rear) {
      cout<<"Queue Underflow ";
      return ;
   } else {
      cout<<"Element deleted from queue is : "<< queue[front] <<endl;
      front++;;
   }
}
void Display() {
   if (front == - 1)
   cout<<"Queue is empty"<<endl;
   else {
      cout<<"Queue elements are : ";
      for (int i = front; i <= rear; i++)
         cout<<queue[i]<<" ";
         cout<<endl;
   }
}
int main() {
   int ch;
   cout<<"1) Insert element to queue"<<endl;
   cout<<"2) Delete element from queue"<<endl;
   cout<<"3) Display all the elements of queue"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter your choice : "<<endl;
      cin>>ch;
      switch (ch) {
         case 1: Insert();
         break;
         case 2: Delete();
         break;
         case 3: Display();
         break;
         case 4: cout<<"Exit"<<endl;
         break;
         default: cout<<"Invalid choice"<<endl;
      }
   } while(ch!=4);
   return 0;
}
```

OUTPUT

```
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
50
Enter your choice :
1
Insert the element in queue :
60
Enter your choice :
3
Queue elements are : 50 60
Enter your choice :
2
Element deleted from queue is : 50
Enter your choice :
3
Queue elements are : 60
Enter your choice :
4
Exit
```

Review Questions/ Exercise:

1. Explain the procedure of Queue implementation using linked list.

   _____

2. Write C++ program to implement the Queue using linked list

   _____

   _____

Name: _____

Roll #: _____

Date: _____

Subject Teacher

Remarks: