Unit VI

6 Queue

6.1 : Basic Concept

Q.1 What is queue ? ☞ **[SPPU : June-22, Marks 3]**

Ans. : Definition : The queue can be formally defined as ordered collection of elements that has two ends named as **front** and **rear**. From the front end one can delete the elements and from the rear end one can insert the elements.

For example :

The typical example can be a queue of people who are waiting for a city bus at the bus stop. Any new person is joining at one end of the queue, you can call it as the rear end. When the bus arrives the person at the other end first enters in the bus. You can call it as the front end of the queue.

Following Fig. Q.1.1 represents the queue of few elements.

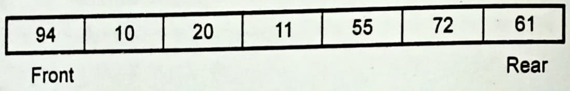


Fig. Q.1.1 Queue

Q.2 Compare Stack and Queue.

Ans. :

Sr. No.	Stack	Queue
1.	The stack is a LIFO data structure. That is the element which is inserted last will be removed first from the stack.	The queue is a FIFO data structure. That means the element which is inserted first will be removed first.
2.	The insertion and deletion of the elements in the stack is done from only one end, called top.	The insertion of the element in the queue is done by the end called rear and the deletion of the element from the queue is done by the end called front.

6.2 : Queue as Abstract Data Type

Q.3 Give an ADT for Queue.

Ans. : The ADT for queue is as given below -

AbstractDataType Queue

{

Instances :

Que[MaX] is a finite collection of elements in which insertion of element is by rear end and deletion of element is by front end.

Precondition :

The front and rear should be within the maximum size MAX.

Before insertion operation, whether the queue is full or not is checked.

Before any deletion operation, whether the queue is empty or not is checked.

Operations :

1. Create() - The queue is created by declaring the data structure for it.
2. insert() - The element can be inserted in the queue by rear end.
3. delet() - The element at front end deleted each time.
4. Display() - The elements of queue are displayed from front to rear.

}

6.3 : Queue Operations

Q.4 Explain the insertion of element in queue implemented using arrays.

Ans. : The insertion of any element in the queue will always take place from the rear end.

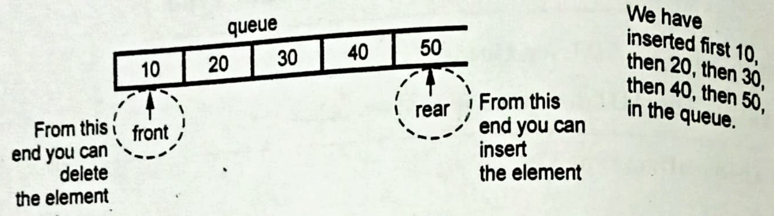

queue
10
20
30
40
50
front
rear
From this end you can delete the element
From this end you can insert the element
We have inserted first 10, then 20, then 30, then 40, then 50, in the queue.

Fig. Q.4.1 Representing the insertion

```
int MyQ::insert(int item)
{
  if(Q.front ==-1)
  Q.front++;
  Q.que[++Q.rear] = item;
  return Q.rear;
}
```

This condition will occur initially when queue is empty

Always increment the rear pointer and place the element in the queue.

Q.5 Explain the deletion of element from queue implemented using arrays.

Ans. : The deletion of any element in the queue takes place by the front end always.

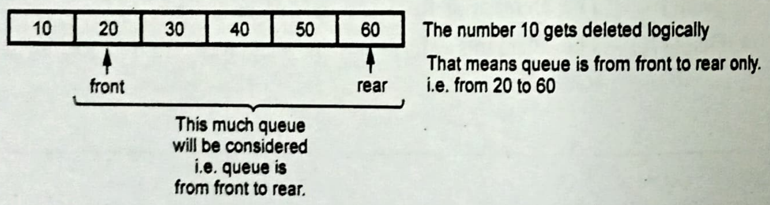
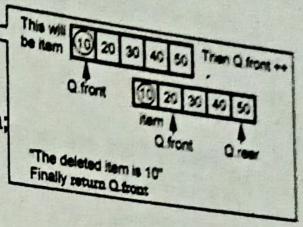

10
20
30
40
50
60
front
rear
This much queue will be considered i.e. queue is from front to rear.
The number 10 gets deleted logically
That means queue is from front to rear only. i.e. from 20 to 60

Fig. Q.5.1 Representing the deletion

```
int MyQ::delet()
{
int item;
item = Q.que[Q.front];
Q.front++;
cout<<"\n The deleted item is "<<item;
return Q.front;
}
```



6.4 : Circular Queue

Q.6 Explain the concept of circular queue.

☞ [SPPU : June-22, Marks 3]

Ans. : Circular queue is a queue in which read and front are adjacent to each other. It can be represented as follows -

For rear and front pointers following formula is used

rear = (rear + 1)%SIZE

front = (front + 1)%SIZE

where SIZE represents the SIZE of a queue.

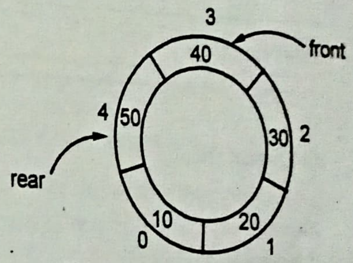


Fig. Q.6.1 Circular queue

Q.7 Implement insert and delete operations of circular queue.

OR Write Pseudo C++ code to implement circular queue using arrays.

☞ [SPPU : June-22, Marks 9]

Ans. :

```
/*
------------------------------------------------
insert function
------------------------------------------------
*/

void Queue::insert(int item)
```

```
{
if (front==(rear+1)%MAX)
{
   cout << "Queue is full\n";
}
else
{
   //setting front pointer for a single element in Queue
   if(front==-1)
   front=rear=0;
   else
   rear=(rear+1)%MAX;
   Que[rear]=item;
}
}
/*
-------------------------------------------------------
delet function
-------------------------------------------------------
*/
int Queue::delet()
{
 int val;
 if(front==-1)
 {
   cout << "Queue is empty\n";
   return 0; // return null on empty Queue
 }
 val= Que[front];//item to be deleted
 if(front==rear)//when single element is present
 {
   front=rear=-1;
 }
 else
   front=(front+1)%MAX;
return val;
}
```

6.5 : Multi-queues

Q.8 Write a short note on - Multiple queues.

Ans. : • One of the application of queues is categorization of data. And multiple queues can be used to store variety of data. We can implement multiple queues using single dimensional arrays.

• In a one dimensional array, multiple queues can be placed. Insertion from its rear end and deletion from its front end can be possible for desired queue. Refer Fig. Q.8.1

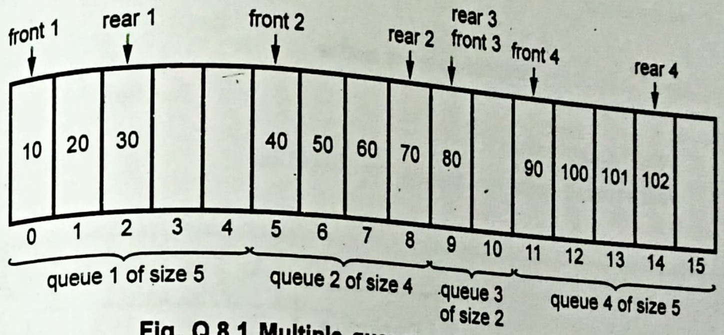


Fig. Q.8.1 Multiple queues using single array

• There are four queues having their own front and rear positioned at appropriate points in a single dimensional array.

• We can perform insertion and deletion of any element for any queue. We can declare the messages "queue full" and "queue empty" at appropriate situations for that particular queue.

6.6 : Linked Queue and Operations

Q.9 Write a routine to insert an element in a linked queue.

☞[SPPU : June-22, Marks 9]

Ans. :

```
void Lqueue::insert()
{
char ch;
Q *temp;
clrscr();
temp =new Q;//allocates memory for temp node
temp->next=NULL;
cout<<"\n\n\n\tInsert the element in the Queue\n";
cin>>temp->data;

if(front == NULL)//creating first node
{
 front= temp;
 rear=temp;
}
else              //attaching other nodes
 {
  rear->next=temp;
  rear=rear->next;
 }
}
```

6.7 : Dequeue

Q.10 Explain the concept of Dequeue with example.

Ans. : Dequeue is a data structure in which we can insert the element both by front and rear end. Similarly we can delete the element from deque by both the rear and front ends.

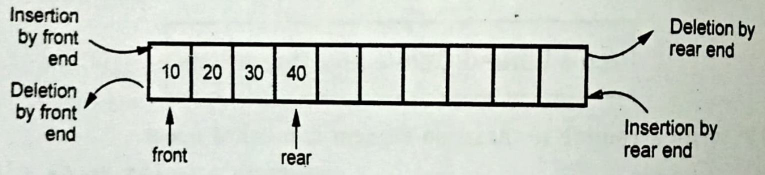
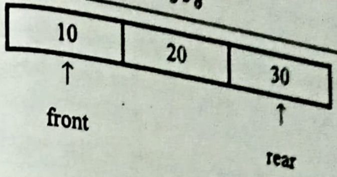


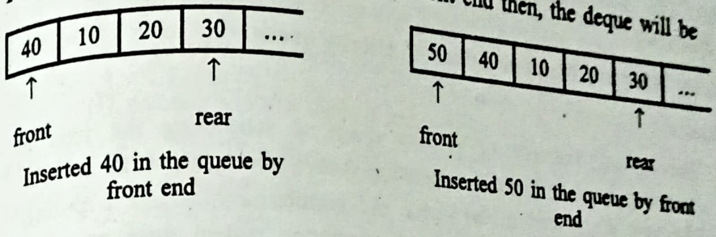
Fig. Q.10.1 Doubly ended queue

As we know, normally we insert the elements by rear end and delete the elements from front end. Let us say we have inserted the elements 10, 20, 30 by rear end.



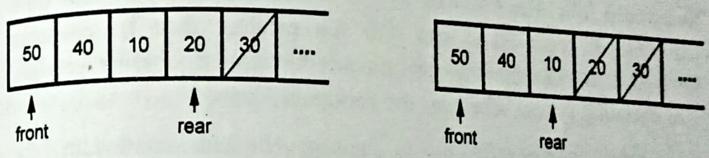
Now if we wish to insert any element from front end then first we have to shift all the elements to the right.

For example if we want to insert 40 by front end then, the deque will be



40
10
20
30
....
front
rear
Inserted 40 in the queue by front end
50
40
10
20
30
...
front
rear
Inserted 50 in the queue by front end

(a) Insertion by front end



(b) Deletion by rear end

Fig. Q.10.2 Operations on deque

We can place –1 for the element which has to be deleted.

6.8 : Priority Queue

Q.11 Write a short note on - priority queue.

☞ [SPPU : June-22, Marks 8]

Ans. : The elements in the priority queue have specific ordering. There are two types of priority queues -

1. **Ascending Priority Queue** - It is a collection of items in which the items can be inserted arbitarily but only smallest element can be removed.

2. **Descending Priority Queue** - It is a collection of items in which insertion of items can be in any order but only largest element can be removed.

In priority queue, the elements are arranged in any order and out of which only the smallest or largest element allowed to delete each time.

The implementation of priority queue can be done using arrays or linked list. The data structure heap is used to implement the priority queue effectively.

Q.12 What are the applications of priority queue.

Ans. :

1. The typical example of priority queue is scheduling the jobs in operating system. Typically. operating system allocates priority to jobs. The jobs are placed in the queue and position1 of the job in priority queue determines their priority. In operating system there are three kinds of jobs. These are real time jobs, foreground jobs and background jobs. The operating system always schedules the real time jobs first. If there is no real time job pending then it schedules foreground jobs. Lastly if no real time or foreground jobs are pending then operating system schedules the background jobs.
2. In network communication, to manage limited bandwidth for transmission the priority queue is used.
3. In simulation modeling, to manage the discrete events the priority queue is used.

END...