QUEUE NOTES

A queue is a FIFO (First-In, First-Out) data structure in which the element that is inserted first is the first one to be taken out. The elements in a queue are added at one end called REAR and removed from another end called FRONT.
Queues can be implemented by using either arrays or linked lists.

ARRAY REPRESENTATION OF QUEUES:

Queues can be easily represented using linear arrays. As stated earlier, every queue has a front and rear end that is represented using pointers, it is the place from where deletions and insertions can be done, respectively.

Operations on Queues -

Before inserting an element in a queue, we must check for overflow conditions. An overflow will occur when we try to insert an element into a queue that is already full. When REAR = MAX – 1, where MAX is the size of the queue, we have an overflow condition. Note that we have written MAX – 1 because the index starts from 0.

```
Step 1:  IF REAR = MAX - 1
             WRITE "OVERFLOW"
             Goto Step 4
         [END OF IF]
Step 2:  IF FRONT = -1 and REAR = -1
             SET REAR = FRONT = 0
         ELSE
             SET REAR = REAR + 1
         [END OF IF]
Step 3:  SET QUEUE[REAR] = NUM
Step 4:  EXIT
```

Similarly, before deleting an element from a queue, we must check for underflow conditions. An underflow condition occurs when we try to delete an element from a queue that is already empty. If FRONT = –1 and REAR = –1, it means there is no element in the queue.

```
Step 1: IF FRONT = -1 OR FRONT > REAR
                WRITE "UNDERFLOW"
        ELSE
                SET VAL = QUEUE[FRONT]
                SET FRONT = FRONT + 1
        [END OF IF]
Step 2: EXIT
```

LINKED REPRESENTATION OF QUEUES:

We have seen how a queue is created using an array. Its drawback is that the array must be declared to have some fixed size. If we allocate space for 50 elements in the queue and it hardly uses 20–25 locations, then half of the space will be wasted. And in case we allocate less memory locations for a queue that might end up growing large and large, re-allocations will consume a lot of time.
If the array size cannot be determined in advance, the other alternative is linked representation.
The storage requirement of linked representation of a queue with n elements is O(n) and the typical time requirement for operations is O(1).
In a linked queue, every element has two parts, one that stores the data and another that stores the address of the next element. The START pointer of the linked list represents FRONT. Another pointer called REAR will store the address of the last element in the queue. All insertions use REAR and all deletions will use FRONT. If FRONT = REAR = NULL, then it indicates that the queue is empty.

Operations on Linked Queues -

A queue has two basic operations: insert and delete. The insert operation adds an element to the end of the queue, and the delete operation removes an element from the front or the start of the queue. Apart from this, there is another operation peek which returns the value of the first element of the queue.

Insert Operation

The insert operation is used to insert an element into a queue. The new element is added as the last element of the queue.
To insert an element, we first check if FRONT=NULL. If the condition holds, then the queue is empty. So, we allocate memory for a new node, store the value in its data part and NULL in its next part. The new node will then be called both FRONT and REAR. However, if FRONT!=NULL, then we will insert the new node at the rear end of the linked queue and name this new node as REAR.

Step 1:	ALLOCATE MEMORY FOR THE NEW NODE AND NAME IT AS PTR
Step 2:	SET PTR DATA = VAL

Step 3:	IF FRONT = NULL
	    SET FRONT = REAR = PTR
	    SET FRONT NEXT = REAR NEXT = NULL
	ELSE
	    SET REAR NEXT = PTR
	    SET REAR = PTR
	    SET REAR NEXT = NULL
	[END OF IF]

Step 4:	END

Delete Operation

The delete operation is used to delete the element that is first inserted in a queue. However, before deleting the value, we must first check if FRONT=NULL because if this is the case, then the queue is empty and no more deletions can be done. If an attempt is made to delete a value from a queue that is already empty, an UNDERFLOW message is printed. If the condition is false, then we delete the first node pointed by FRONT. The FRONT will now point to the second element of the linked queue.

Step 1:	IF FRONT = NULL
	    Write Underflow

	Go to Step 5 [END OF IF]
Step 2:	SET PTR = FRONT
Step 3:	SET FRONT = FRONT -> NEXT
Step 4:	FREE PTR
Step 5:	END

TYPES OF QUEUES

A queue data structure can be classified into the following types:

1. Linear Queue
2. Circular Queue
3. Priority Queue

Circular Queues

In linear queues, we have discussed so far that insertions can be done only at one end called the REAR and deletions are always done from the other end called the FRONT.
Now, if you want to insert another value, it will not be possible because the queue is completely full. There is no empty space where the value can be inserted.
Consider a scenario in which two successive deletions are made. Suppose we want to insert a new element in the queue. Even though there is space available, the overflow condition still exists because the condition REAR = MAX – 1 still holds true. This is a major drawback of a linear queue.
To resolve this problem, we have two solutions. First, shift the elements to the left so that the vacant space can be occupied and utilized efficiently. But this can be very time-consuming, especially when the queue is quite large.
The second option is to use a circular queue. In the circular queue,the last pointer points to the first pointer of the queue. The circular queue will be full only when FRONT = 0 and REAR = MAX – 1. A circular queue is implemented in the same manner as a linear queue is implemented. The only difference will be in the

code that performs insertion and deletion operations.

For insertion, we now have to check for the following three conditions:
If FRONT = 0 and REAR = MAX – 1, then the circular queue is full. Look at the queue given in Fig. 8.16 which illustrates this point.
If REAR!= MAX – 1, then rear will be incremented and the value will be inserted as illustrated in Fig. 8.17.
If FRONT!= 0 and REAR = MAX – 1, then it means that the queue is not full. So, set REAR = 0 and insert the new element there.

Step 1:	IF FRONT = and Rear = MAX-1
	        Write OVERFLOW
	        Goto step 4
	[END OF IF]

Step 2:	IF FRONT=-1 and REAR=-1
	        SET FRONT = REAR = 0
	ELSE IF REAR = MAX-1 and FRONT != 0
	        SET REAR = 0
	ELSE
	        SET REAR = REAR+1
	[END OF IF]

Step 3:	SET QUEUE[REAR] = VAL
Step 4:	EXIT