2. Implementation of Priority Queue

AIM: Implement priority queue as ADT using single linked list for servicing patients in an hospital with priorities as i) Serious (top priority) ii) medium illness (medium priority) iii) General (Least priority)

OBJECTIVE:

1) To understand the concept of priority Queue.
2) How data structures Queue is represented as an ADT.

THEORY:

1) What is Queue? Explain Queue operations with neat diagrams?

A queue is a particular kind of collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once an element is added, all elements that were added before have to be removed before the new element can be invoked. A queue is an example of a linear data structure.

Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer. Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes. Common implementations are circular buffers and linked lists. Queue is a data structure that maintain "First In First Out" (FIFO) order. And can be viewed as people queueing up to buy a ticket. In programming, queue is usually used as a data structure for BFS (Breadth First Search).

Operations on queue:

1. enqueue - insert item at the back of queue Q
2. dequeue - return (and virtually remove) the front item from queue Q
3. displayfront - return (without deleting) the front item from queue Q
4. displayrear - return (without deleting) the reart item from queue Q

2) Explain how Queue can be implemented as an ADT.

Theoretically, one characteristic of a queue is that it does not have a specific capacity. Regardless of how many elements are already contained, a new element can always be

added. It can also be empty, at which point removing an element will be impossible until a new element has been added again. A practical implementation of a queue, e.g. with pointers, of course does have some capacity limit, that depends on the concrete situation it is used in. For a data structure the executing computer will eventually run out of memory, thus limiting the queue size. Queue overflow results from trying to add an element onto a full queue and queue underflow happens when trying to remove an element from an empty queue. A bounded queue is a queue limited to a fixed number of items.

3) What is Priority Queue ? Explain with example.

priority queue is an abstract data type in computer programming that supports the following three operations:

- insertWithPriority: add an element to the queue with an associated priority
- getNext: remove the element from the queue that has the highest priority, and return it (also known as "PopElement(Off)", or "GetMinimum")
- peekAtNext (optional): look at the element with highest priority without removing it.

The rule that determines who goes next is called a queueing discpline. The simplest queueing discipline is called FIFO, for "first-in-first-out." The most general queueing discipline is priority queueing, in which each customer is assigned a priority, and the customer with the highest priority goes first, regardless of the order of arrival. The reason I say this is the most general discipline is that the priority can be based on anything: what time a flight leaves, how many groceries the customer has, or how important the customer is. Of course, not all queueing disciplines are "fair," but fairness is in the eye of the beholder.

The Queue ADT and the Priority Queue ADT have the same set of operations and their interfaces are the same. The difference is in the semantics of the operations: a Queue uses the FIFO policy, and a Priority Queue (as the name suggests) uses the priority queueing policy. As with most ADTs, there are a number of ways to implement queues Since a queue is a collection of items, we can use any of the basic mechanisms for storing collections: arrays, lists, or vectors. Our choice among them will be based in part on their performance--- how long it takes to perform the operations we want to perform--- and partly on ease of implementation.

ALGORITHM:

Define structure for Queue(Priority, Patient Info, Next Pointer).

Empty Queue:

Return True if Queue is Empty else False.
isEmpty(Front)
Front is pointer of structure,which is first element of Queue.
Step 1: If Front = = NULL
Step 2: Return 1

Step 3: Return 0

Insert Function:

Insert Patient in Queue with respect to the Priority.
Front is pointer variable of type Queue,which is 1st node of Queue.
Patient is a pointer variable of type Queue, which hold the information about new patient.

```
Insert(Front, Queue )
    Step 1: If Front = = NULL //Queue Empty
              Then  Front = Patient;
    Step 2: Else if Patient->Priority >  Front->Priority
              Then i) Patient->Next = Front;
             ii) Front=Patient;
    Step 3: Else A) Temp = Front;
                 B)   Do Steps a while Temp != NULL And
                          Patient->Priority <= Temp->Next->Priority
                                a)  Temp=Temp->Next;
                 c) Patient->Next = Temp->Next;
                      Temp->Next = Patient;
    Step 4: Stop.
```

Delete Patient details from Queue after patient get treatment:

Front is pointer variable of type Queue,which is 1st node of Queue.
Delete Node from Front.

```
Delete( Front )
Step 1: Temp = Front;
Step 2: Front = Front->Next;
Step 3: return Temp
```

Display Queue Front:

Front is pointer variable of type Queue,which is 1st node of Queue.

```
Display( Front )
Step 1: Temp = Front;
Step 2: Do Steps  while Temp != NULL
        a) Display Temp Data
        b)        If  Priority 1 Then “General Checkup”;
          Else If Priority 2 Then Display “ Non-serious";
          Else If Priority 3 Then Display "Serious"
                   Else Display "Unknown";
           c) Temp = Temp->Next;
Step 3: Stop.
```

Display Queue rear:

Front is pointer variable of type Queue,which is 1st node of Queue.

```
Display( Rear )
Step 1: Temp = Rear;
Step 2: Do Steps  while Temp != NULL
        a) Display Temp Data
        b)        If  Priority 1 Then “General Checkup”;
```

Else If Priority 2 Then Display " Non-serious";
Else If Priority 3 Then Display "Serious"
Else Display "Unknown";
c) Temp = Temp->Next;

Step 3: Stop.

INPUT:

Test Case	O/P
Queue Empty	Display message "Queue Empty"
Queue Full	Display message "Queue Full"

Name of patients & category of patient like

a) Serious (top priority), b) non-serious (medium priority), c) General Checkup (Least priority).

E.g. Enter patient arrival in the hospital with following priorities 1, 3, 2, 2, 1, 3

OUTPUT:

Priority queue cater services to the patients based on priorities.
Patient should be given service in the following order 3, 3, 2, 2, 1, 1

Note: 3 means top priority.

FAQ:

1. What are the types of data structure?
2. What are the operations can implement on queue?
3. What is circular queue?
4. What is Multiqueue?
5. Explain importance of stack in recursion
6. Explain Implicit & Explicit Stack.
7. Applications of stack and Queue as a data structure