Lab-10

Heap Tree as Priority Queue

Objective:
This lab will provide you with the inside knowledge of another very important type of Queue, i.e. Priority Queues.

Activity Outcomes:
• Design Priority Queue Array based
• Design Priority Queue Dynamic
• Apply priority Queue in a real-life Example

- **Useful Concepts**
  Till now you have studied that inn a Queue we go on with the first in first out system but what to do if you encounter such a system where this system partially fails, i.e. you require the same system of first out but the incoming queue is settled according to a defined system??
- We have studied that there is a special type of Queue in which we can edit form both the ends, i.e. DEQUE.
- For a better understanding of this topic consider a Hospital System where we first serve the patients who are brought in a critical condition. We leave all the patients with not so serious issues and the Doctor give the treatment at a priority to the one brought in critical condition. Hence, we can grasp an idea that every element comes with its own priority, and are served in accordance. This is the same idea we will implement for a Priority Queue.

Activity 1:
Design an Array Based Priority Queue. Write the enqueue method.

```
private int[] H = new int[50];
private int size = -1;

// Function to return the index of the parent node of a
given node
    private int parent(int i) {
        return (i - 1) / 2;
    }

// Function to shift up the node in order to maintain
the heap property
    private void shiftUp(int i) {
        while (i > 0 && H[parent(i)] < H[i]) {
            // Swap parent and current node
            swap(H, parent(i), i);
            // Update i to parent of i
            i = parent(i);
        }
    }
```

```
public void insert(int p) {
        size = size + 1;
        H[size] = p;
        // Shift Up to maintain heap property
        shiftUp(size);
    }
```

Activity 2:

Dequeue the element from queue

```
// Function to extract the element with maximum
priority
int extractMax()
    {
        int result = H[0];
        // Replace the value at the root
        // with the last leaf
        H[0] = H[size];
        size = size - 1;
        // Shift down the replaced element
        // to maintain the heap property
        shiftDown(0);
        return result;
    }
```

```
// Function to shift down the node in
// order to maintain the heap property
    void shiftDown(int i)
    {
        int maxIndex = i;
        // Left Child
        int l = leftChild(i);
        if (l <= size && H[l] > H[maxIndex]) {
            maxIndex = l;
        }
        // Right Child
        int r = rightChild(i);
        if (r <= size && H[r] > H[maxIndex]) {
            maxIndex = r;
        }
        // If i not same as maxIndex
        if (i != maxIndex) {
            swap(H[i], H[maxIndex]);
            shiftDown(maxIndex);
        }
    }
```

```
// Function to return the index of the left child of
the given node
    int leftChild(int i)
    {
        return ((2 * i) + 1);
    }

// Function to return the index of the right child of
the given node
    int rightChild(int i)
    {
        return ((2 * i) + 2);
    }
```

Lab Task 1

Design a Hospital Registration system for patients using Priority Queue.

a. Assume an *interface* that you think is appropriate for such a system.

b. Take the priority factor, the condition of patient.

c. Design it using the above implemented system of Priority Queue.