Data Structures Lab (CSC204) **Session: Monsoon 2021-22**
III BTech (CSE) **Date: 05.10.2021**

Assignment-7 (Tree)

1. A tree is a hierarchical data structure that stores information in different levels. A binary search tree is a common type of tree which contains at most two children. We refer these two children as left and right child. A node in binary search tree typically consists of three fields namely, pointer to left child, data, and pointer to right child. In binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root and the value of all the nodes in the right sub-tree is greater than or equal to the value of the root. In this problem, your task is to write a program to create a binary search tree based on the given data. The tree can be traversed in different ways such as *pre-order*, *in-order*, *post-order*, and *level order* traversal. So, you are required to implement these traversals and display the nodes of the created tree based on these four traversals.

   Sample Input:
   $Enter\ the\ node\ values\ for\ tree = 25, 34, 20, 15, 24, 30, 40$

   Sample Output:
   $The\ pre - order\ traversal\ of\ the\ tree\ is$: $25, 20, 15, 24, 34, 30, 40$
   $The\ in - order\ traversal\ of\ the\ tree\ is$: $15, 20, 24, 25, 30, 34, 40$
   $The\ post - order\ traversal\ of\ the\ tree\ is$: $15, 24, 20, 30, 40, 34, 25$
   $The\ level\ order\ traversal\ of\ the\ tree\ is$: $25, 20, 34, 15, 24, 30, 40$

2. Suppose you are given an array of integers. Your task is to find out k largest elements of the array. To perform this task, you are required to build a max-heap from the array elements. You can delete the root element from the heap to get the largest element. In this way, you can obtain *k* largest elements from the *max-heap* by extracting elements *one by one*. After deletion of each node, you need to apply $heapify()$ function on the heap to maintain max-heap property.

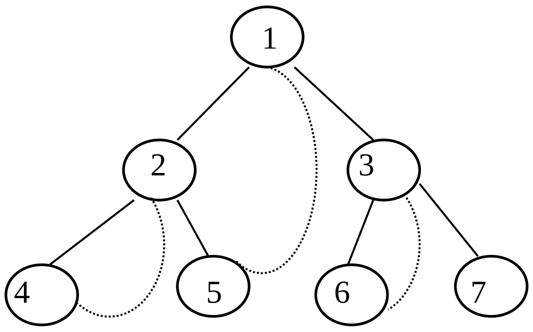
   Sample Input:
   $Enter\ the\ array\ elements =\ 5, 7, 9, 10, 12, 15, 20$
   $Enter\ the\ value\ of\ k =\ 3$

   Sample Output:
   $k\ largest\ elements\ are =\ 20, 15, 12$

3. The *in-order* traversal of binary tree can be performed using recursion or auxiliary stack. In this problem, you are required to utilize threaded binary tree for finding *in-order* traversal in an efficient way. You are required to construct a single threaded binary tree in which the right pointers of nodes create threads. The right pointers of node, which are *NULL*, point to the *in-order* successor of the node. You can utilize the following procedure for finding *in-order* traversal of the threaded tree:

    I. Find the leftmost node in the tree. Mark this node as current. This node will be the first node in the traversal.
    II. Repeat steps III-IV until the current node is not *NULL*.
    III. If current node contains a thread, then go to right pointer of current.
    IV. Else find the leftmost child in the right sub-tree of current node. Mark this node as current.

    Sample Input:



Sample Output:

$In - order\ traversal\ of\ tree\ is: 4, 2, 5, 1, 6, 3, 7$