National Institute of Technology, Calicut
Department of Computer Science and Engineering
CS2094 – Data Structures Lab

Assignment 4 (Advanced)

Submission deadline (on or before): **17.03.2015 (Tuesday) 5pm**

Naming Conventions for submission

Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar or .tar.gz). The name of this file must be ASSG4A<Number>_<ROLLNO>_<FIRSTNAME>.zip (For example: ASSG4A_BxxyyyyCS_LAXMAN.zip). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

The source codes must be named as ASSG4A<Number>_<ROLLNO>_<FIRST NAME>_<PROGRAM-NUMBER>.<extension> (For example: ASSG4A_BxxyyyyCS_ LAXMAN_1.c).

Questions

1. Define M, the size of the largest BST, of a binary tree, as the maximum number of nodes in a binary search tree that is a subtree of the binary tree. To cite a few examples: M(null) = 0, M(T)=1 if T is a single node tree, and M(T) = |T|, if T is a binary search tree. Hence if T is not a binary search tree 1<= M(T) < |T|. (Note:|T|= number of nodes in T)

Design and implement an O(n) time, recursive algorithm, to find M(T), given T.

In your design, apart from data nodes as per your design if required, there should be only two pointers, the left and the right, to the left and right child. You may make the following assumptions:

- Keys for the nodes are positive integers.
- All keys in the list need not be unique.
- The input file is in ascii text form in the following format:
  a. First line contains the number of nodes in the binary tree
  b. Second line contains the list of key values in array format for representing a binary tree. NULL is represented by -1.

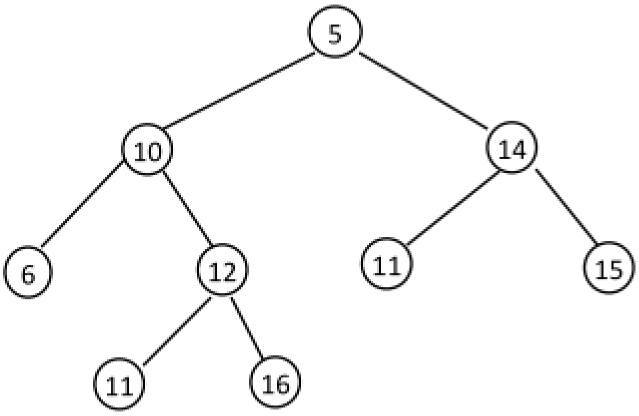
Output Specifications:

- o After creating the tree, first print the inorder traversal of the nodes in the tree
- o Then, print the size of the largest BST.

Sample Input:

```
9
5  10  14  6  12  11  15  -1  -1  11  16  -1  -1  -1  -1
```

The figure below describes the binary tree given as input.

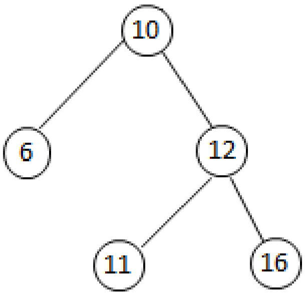


Output for the sample input

Inorder Traversal: 6 10 11 12 16 5 11 14 15

Maximum size of BST = 5

(Explanation: Largest subtree which is BST, is shown below)



2. Given a binary search tree T, design and implement a recursive algorithm to create a singly linked list of nodes present at each depth of T in O(n) time. In your design, apart from data nodes as per your design if required, there should be only two pointers, the left and the right, to the left and right child.
Note: The depth of a node in a tree is the number of edges in the path from the root to that node. Hence the depth of root node is 0. The depth of a tree is the maximum depth of a leaf in the tree. Hence, the number of linked lists created should be greater by one than the depth of the given binary tree.
You may make the following assumptions

- Keys for the nodes are positive integers.
- All keys in the list are unique.
- The input file is in ascii text form in the following format:
    a. First line contains number of nodes in the tree
    b. Second line contains the list in array format for representing the tree. In that, -1 represents Null.

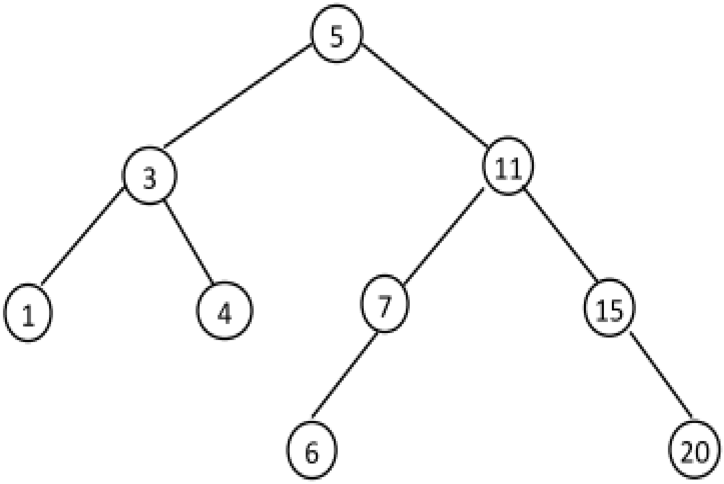
Output Specifications:

- After creating the tree, first print the inorder traversal of the nodes in the tree
- Then, keys in each different linked list are printed, separated by new line characters.
- The nodes within a same list are separated by '→' marker.

Sample Input:

9 5 3 11 1 4 7 15 -1 -1 -1 -1 6 -1 -1 20

(The input tree is described by the following figure)



Sample Output:

```
Inorder Traversal: 1  3  4  5  6  7  11  15  20

5
3 → 11
1 → 4 → 7 → 15
6 → 20
```

Explanation: The depth of the given binary search tree is 3. Hence 4 singly linked lists are created where each linked list contains nodes at same depth.

3. Given the preorder traversal of a binary search tree with unique keys, design and implement an algorithm to create the binary search tree and print it in a scheme like format.

Sample Input:

43 15 8 30 20 35 60 50 82 70

Sample Output:

(43 (15 (8 ( ) ( )) (30 (20 ( ) ( )) (35 ( ) ( )))) (60 (50 ( ) ( )) (82 (70 ( ) ( )) ( ))))