

**National Institute of Technology Calicut**  
**Department of Computer Science and Engineering**  
**Fourth Semester B. Tech.(CSE)-Winter 2021-22**  
**CS2094D Data Structures Laboratory**  
**Assignment #1**

**Submission deadline (on or before):** 10.01.2022, 9:00 AM

**Policies for Submission and Evaluation:**

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

**Naming Conventions for Submission**

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

**ASSG<NUMBER>\_<ROLLNO>\_<FIRST-NAME>.zip**

(Example: *ASSG1\_BxyyyyyCS\_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

**ASSG<NUMBER>\_<ROLLNO>\_<FIRST-NAME>\_<PROGRAM-NUMBER>.c**

(For example: *ASSG1\_BxyyyyyCS\_LAXMAN\_1.c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

**Standard of Conduct**

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: [http://cse.nitc.ac.in/sites/default/files/Academic-Integrity\\_new.pdf](http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf).

## QUESTIONS

1. A BINARY TREE is a rooted tree in which each node is an object that contains a *key* value. In addition to a *key* and satellite data, each node contains attributes *left*, *right*, and *p* that point to the nodes corresponding to its left child, its right child, and its parent, respectively. If a child or the parent is missing, the appropriate attribute contains the value NIL. The root node is the only node in the tree whose parent is NIL. The PARENTHESIS REPRESENTATION of a binary tree is recursively defined as given below:

- The string `( )` represents an empty tree.
- The string `( k left-subtree right-subtree )` represents a tree whose root node has key *k*, left-subtree is the left subtree of the root node in PARENTHESIS REPRESENTATION and right-subtree is the right subtree of the root node in PARENTHESIS REPRESENTATION.

Write a program to create a binary tree *T* and print *T* in PARENTHESIS REPRESENTATION. Your program should contain the following functions:

- INSERT(*T*, *k*) - inserts the element *k* to the tree *T*.  
**Note:** To insert an element *k* into a binary tree *T*, do a level order traversal of the given tree *T*. If we find a node whose left child is empty, make new key *k* as left child of the node. Else if we find a node whose right child is empty, make the new key *k* as right child. i.e., keep traversing the tree until we find a node whose either left or right child is empty.
- PRINT(*T*) - that should take as input a pointer to the root node of a binary tree and print the tree in its PARENTHESIS REPRESENTATION.

### Input format:

- Each line contains a character from '*i*', '*p*' and '*e*' followed by at most one integer. The integers, if given, are in the range  $[-10^6, 10^6]$ .
- Character '*i*' is followed by an integer separated by single space. In this operation, a node with this integer as key is created and inserted into *T*.
- Character '*p*' is to print the PARENTHESIS REPRESENTATION of the tree *T*.
- Character '*e*' is to 'exit' from the program.

### Output Format:

- The output is the space separated PARENTHESIS REPRESENTATION of the tree *T*.

### Sample Input:

```
i 4
i 3
i 9
i 5
i 6
i 1
i 8
p
e
```

### Sample Output:

```
( 4 ( 3 ( 5 ( ) ( ) ) ( 6 ( ) ( ) ) ) ( 9 ( 1 ( ) ( ) ) ( 8 ( ) ( ) ) ) )
```

2. The BINARY SEARCH TREE (BST) data structure supports many of the dynamic-set operations. A BST is organized as a binary tree in which each node is an object that contains a *key* value. In addition to a *key* and satellite data, each node contains attributes *left*, *right*, and *p* that point to the nodes corresponding to its left child, its right child, and its parent, respectively. If a child or the parent is missing, the appropriate attribute contains the value NIL. The root node is the only

node in the tree whose parent is NIL. The keys in a binary search tree are always stored in such a way as to satisfy the **binary-search-tree** property:

- Let  $x$  be a node in a binary search tree. If  $y$  is a node in the left subtree of  $x$ , then  $y.key \leq x.key$ . If  $y$  is a node in the right subtree of  $x$ , then  $y.key \geq x.key$ .

Write a program to create a BINARY SEARCH TREE  $T$  and perform the operations *insertion*, *deletion*, *search*, *find level*, *find minimum*, *find maximum*, *predecessor*, *successor* and *traversals* (inorder, preorder and postorder) on  $T$ . Input should be read from console and output should be shown in console. Your program should include the following functions:

- MAIN() - creates the Binary Search Tree  $T$  with  $T$  as the root node (which is NIL initially) and repeatedly reads a character 'a', 'd', 's', 'l', 'm', 'x', 'r', 'u', 'i', 'p' or 't' from the console and calls the sub-functions appropriately until character 'e' is entered for exiting the program.
- CREATENODE( $k$ ) creates a new node with *key* value  $k$  and returns a pointer to the new node. All the pointer attributes of the new node are set to NIL.
- INSERT( $T, x$ ) - inserts the node  $x$  into the BST  $T$ .  
**Note:** The caller of this function is assumed to create the node  $x$  using the CREATENODE function.
- DELETE( $T, x$ ) - deletes the node  $x$  from the BST  $T$ .  
**Note:** The caller of this function is assumed to invoke SEARCH function to locate the node  $x$ .
- SEARCH( $T, k$ ) - searches for a node with key  $k$  in  $T$ , and returns a pointer to a node with key  $k$  if one exists; otherwise, it returns NIL.
- LEVEL( $T, k$ ) - searches for a node with key  $k$  in  $T$ , and returns the level of the node with key  $k$  if one exists; otherwise, it returns NIL.
- MINVALUE( $T$ ) returns the minimum value in the BST  $T$ .
- MAXVALUE( $T$ ) returns the maximum value in the BST  $T$ .
- PREDECESSOR( $T, y$ ) - searches for a node with key  $y$  in  $T$ , and returns a pointer to a node which is predecessor of the node with key  $y$  if one exists; otherwise, it returns NIL.
- SUCCESSOR( $T, y$ ) - searches for a node with key  $y$  in  $T$ , and returns a pointer to a node which is successor of the node with key  $y$  if one exists; otherwise, it returns NIL.
- INORDER( $T$ ) - performs recursive inorder traversal of the BST  $T$  and prints the key in the nodes of  $T$  in inorder.
- PREORDER( $T$ ) performs recursive preorder traversal of the BST  $T$  and prints the key in the nodes of  $T$  in preorder.
- POSTORDER( $T$ ) performs recursive postorder traversal of the BST  $T$  and prints the key in the nodes of  $T$  in postorder.

#### Input format:

- Each line contains a character from 'a', 'd', 's', 'l', 'm', 'x', 'r', 'u', 'i', 'p', 't' or 'e' followed by at most one integer. The integers, if given, are in the range  $[-10^6, 10^6]$ .
- Character 'a' is followed by an integer separated by single space. In this operation, a node with this integer as key is created and inserted into  $T$ .
- Character 'd' is followed by an integer separated by single space. In this operation, the node with this integer as key is deleted from  $T$  and the deleted node's key is printed.
- Character 's' is followed by an integer separated by single space. This operation is to find the node with this integer as key in  $T$ .
- Character 'l' is followed by an integer separated by single space. This operation is to find the level of the node with this integer as key in  $T$ .
- Character 'm' is to find the minimum value of  $T$ .
- Character 'x' is to find the maximum value of  $T$ .

- Character ‘*r*’ is followed by an integer separated by single space. This operation is to find the predecessor of the node with this integer as key in  $T$ .
- Character ‘*u*’ is followed by an integer separated by single space. This operation is to find the successor of the node with this integer as key in  $T$ .
- Character ‘*i*’ is to perform inorder traversal of  $T$ .
- Character ‘*p*’ is to perform preorder traversal of  $T$ .
- Character ‘*t*’ is to perform postorder traversal of  $T$ .
- Character ‘*e*’ is to ‘exit’ from the program.

**Output Format:**

- The output (if any) of each command should be printed on a separate line.
- For option ‘*d*’, print the deleted node’s key. If a node with the input key is not present in  $T$ , then print -1.
- For option ‘*s*’, if the key is present in  $T$ , then print 1. If key is not present in  $T$ , then print -1.
- For option ‘*l*’, if the key is present in  $T$ , then print its level. If key is not present in  $T$ , then print -1.
- For option ‘*m*’, print the minimum value of  $T$ .
- For option ‘*x*’, print the maximum value of  $T$ .
- For option ‘*r*’, if the key is present in  $T$ , then print its predecessor. If key is not present in  $T$ , then print -1.
- For option ‘*u*’, if the key is present in  $T$ , then print its successor. If key is not present in  $T$ , then print -1.
- For option ‘*i*’, print the key in the nodes of  $T$  obtained from inorder traversal.
- For option ‘*p*’, print the key in the nodes of  $T$  obtained from preorder traversal.
- For option ‘*t*’, print the key in the nodes of  $T$  obtained from postorder traversal.

**Sample Input:**

```

a 25
a 13
a 50
a 45
a 55
a 18
l 50
l 19
m
x
r 25
u 30
u 25
i
p
t
s 10
s 25
d 55
d 13
d 10
d 25
i
s 25

```

e

**Sample Output:**

```
2
-1
13
55
18
-1
45
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25
-1
1
55
13
-1
25
18 45 50
-1
```

3. Given the PARENTHESIS REPRESENTATION of a binary tree  $B$ , find the minimum number of *proper subtrees* of  $B$  that are to be deleted from  $B$  to convert it into a Binary Search Tree. A *proper subtree* of a tree  $T$  is a tree consisting of a node  $N$  (different from the root) and all descendants of  $N$  in  $T$ . In particular, a leaf node in  $T$  is a subtree of  $T$ . Note that in a single proper subtree deletion operation, all the nodes in that subtree are deleted.

**Input format:**

- The first and only line of input contains the PARENTHESIS REPRESENTATION of the given binary tree.

**Output format:**

- A single line containing the minimum number of proper subtrees to be deleted from the given binary tree to convert it into a BST.

**Sample Input 1:**

```
( 15 ( 12 ( 32 ( 10 ( ) ( ) ) ( 35 ( ) ( ) ) ) ( 45 ( 41 ( ) ( ) ) ( 60 ( ) ( ) ) ) ) ( 30 ( ) ( 46 ( ) ( 50 ( ) ( ) ) ) ) )
```

**Sample Output 1:**

```
1
```

**Sample Input 2:**

```
( 10 ( 21 ( ) ( ) ) ( 9 ( ) ( ) ) )
```

**Sample Output 2:**

```
2
```

4. Given the PARENTHESIS REPRESENTATION of a binary tree  $T$ . Write a program to find the lowest common ancestor of two nodes in a given binary tree. *Lowest Common Ancestor(LCA)* of two nodes  $u$  and  $v$  in a tree is the shared ancestor of  $u$  and  $v$  that is located farthest from the root.

**Input format:**

- The integers, if given, are in the range  $[-10^6, 10^6]$ .

- First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree  $T$ .
- Second line contains two integers separated by a space, represents the value of  $u$  and  $v$  respectively.

**Output format:**

- The output consist of a single integer value **LCA**.

**Sample Input 1:**

```
( 1 ( 2 ( 4 ( 8 ( ) ( ) ) ( 9 ( ) ( ) ) ) ( 5 ( ) ( ) ) ) ( 3 ( 6 ( 10 ( ) ( ) ) ( ) ( ) ) ) )
5 9
```

**Sample Output 1:**

2

**Sample Input 2:**

```
( 1 ( 2 ( 4 ( 8 ( ) ( ) ) ( 9 ( ) ( ) ) ) ( 5 ( ) ( ) ) ) ( 3 ( 6 ( 10 ( ) ( ) ) ( ) ( ) ) ) )
6 3
```

**Sample Output 2:**

3

5. A sub-tree of a tree  $T$  is a tree  $S$  consisting of a node in  $T$  and all of its descendants in  $T$ . Given a Binary Tree  $T$ , write a program to find the *size* of the largest sub-tree which is a BST where *size* of the BST is the number of nodes present in it.

**Input format:**

- The integers, if given, are in the range  $[-10^6, 10^6]$ .
- Input contains space separated PARENTHESIS REPRESENTATION of the tree  $T$ .

**Output format:**

- Single line representing *size* of the **largest** BST.

**Sample Input 1:**

```
( 10 ( 15 ( 12 ( ) ( ) ) ( 20 ( ) ( ) ) ) ( 8 ( 5 ( ) ( ) ) ( 2 ( ) ( ) ) ) )
```

**Sample Output 1:**

3

**Sample Input 2:**

```
( 10 ( 16 ( 7 ( ) ( ) ) ( 24 ( 19 ( ) ( ) ) ( 29 ( ) ( ) ) ) ) ( 5 ( ) ( ) ) )
```

**Sample Output 2:**

5