Birla Institute of Technology & Science, Pilani
2nd Semester 2016-17 - CS F211 – Data Structures and Algorithms

---

Lab 9 – 01st April 2017
Topics – Binary Search Trees & In-Order Traversal

Problem 1

Implement a Binary Search Tree (BST) along with the following operations it supports –

- *Add*: Inserts a given key *k* into the given BST.
- *Construct*: Calls insert operation repeatedly for a given list of elements one after the other to construct a fresh BST called *bst1*.
- Recursive-InOrderTraversal: Performs in order traversal on the given BST, which is implemented using recursion.
- Iterative-InOrderTraversal: Performs in order traversal on the given BST, which is implemented using iterations, by eliminating the recursive call in the above implementation by using an explicit stack.
- Find-k^{th}-smallest element: Modify Iterative InOrderTraversal to stop traversing when it finds kth smallest element in the tree.
- Find elements between *k1* and *k2*: Modify Iterative InOrderTraversal to finds all the elements in the tree that are lying between the keys *k1* and *k2*.

You can use the following table to design your functions. You can also refer to the sample input and output case provided.

Key	Function	Input Format	Description
0	*readData*	0 *N* k_1 k_2 k_3 ... k_N	Reads the next *N* lines containing *N* keys and stores them into an array of size *N* called *Arr*.
1	*add*	1 *k*	Inserts the given key *k* into the given BST.
2	*construct*	2	Initializes an empty BST *bst1* and inserts all the elements of *Arr* into it, in the same order as they were inserted into *Arr*.
3	*inOrderTravRec*	3	Performs an in order traversal of the tree (implement using recursion) and prints all the nodes in the order they were visited during the traversal. Printing must be in a single line space separated.
4	*inOrderTravIter*	4	Performs an in order traversal of the tree (implemented using iteration by using an explicit stack to eliminate non-tail recursion) and prints all the nodes in the order they were visited during the traversal. Printing must be in a single line space separated.
5	*findkthSmallest*	5 *k*	Finds kth smallest element in the tree by using a ***modified version 1*** of iterative-in-order-traversal and prints it. You need to implement this modified version as required.
6	*findBetweenKeys*	6 *k1 k2*	Finds all the elements in the tree which will lie between *k1* and *k2*. Use a ***modified version 2*** of iterative-in-order-traversal to do this task. Prints all the keys found in a single line space separated. You need to implement this modified version as required.

Sample input and output - 1

Sample Input	Sample Output
0 4 40 23 35 43 2 3 4 5 2 6 10 25 -1	23 35 40 43 23 35 40 43 35 23