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

Lab 9 – 28th March 2017
Topics – Binary Search Trees

Problem 1

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

- *Find*: Finds whether a given key *k* is present a given BST.
- *Add*: Inserts a given key *k* into the given BST.
- *Delete*: Deletes a given key *k* from a given BST. This operation first finds the node containing the given key, *k*, swaps the contents with node containing the in-order successor, say, *n*, and then deletes *n*.
- *FindHeight*: Computes the height of a given BST.
- *Construct*: Calls insert operation repeatedly for a given list of elements one after the other to construct a fresh BST called *bst1*.
- *RandomConstruct*: Picks up elements at a random order from a given list and inserts them into a fresh BST called *bst2*, until all elements in the list are inserted into *bst2*. You must not repeat insertion of any of the elements in the list.

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 BST_Ptr*	Inserts the given key *k* into the given BST being referenced *BST_Ptr*.
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	*randomConstruct*	3	Initializes an empty BST *bst2*. Then starts picking up random elements from *Arr* and inserts them into *bst2*, until all of them are inserted. You must insert each element of *Arr* into the *bst2* only once.
4	*find*	4 *k BST_Ptr*	Finds whether a given key *k* exists in the BST referenced by *BST_Ptr*. Prints 1 if found, 0 otherwise.
5	*delete*	5 *k BST_Ptr*	Deletes the node containing a given key *k* if found in given BST referenced by *BST_Ptr*. Returns 0 otherwise. This function must print the key *k* if deleted from the given BST or print 0 if key not found.
6	*findHeight*	6 *BST_Ptr*	Finds the height of the BST pointed by *BST_Ptr* and prints it.
7	*experiment*	7	Finds the height of *bst1* and *bst2* and prints them.

Sample input and output - 1

Sample Input	Sample Output
0 4 40 23 35 43 2 3 4 33 bst1 4 35 bst2 5 33 bst1 5 35 bst1 6 bst1 6 bst2 7 -1	0 1 0 33 <X> <Y> Height of bst1 is <Y> Height of bst2 is <X>