# Assignment 9: Binary search trees

1. Read n ints and make a binary search tree (BST). Do k search operations to print results as y/n.
   **Input: (n, x_i, k, y_i)**
   ```
   4
   2 1 4 3
   3
   3 7 1
   ```
   **Output:**
   ```
   y
   n
   y
   ```

2. Read n ints and make a BST in the same order. Print the tree in preorder, inorder and postorder traversals. Separate characters by '_'.
   **Input: (n, x_i)**
   ```
   4
   2 1 4 3
   ```
   **Output:**
   ```
   2_1_4_3_
   1_2_3_4_
   1_3_4_2_
   ```

3. Read 2n `int`s. Use each half to create two BSTs in the given order. Find if the two trees are identical. Print y/n. There are T test cases.
   **Input: (T, n, x_i)**
   ```
   3
   3
   1 2 3 1 3 2
   1 2 3 2 3 1
   2 1 3 2 3 1
   ```
   **Output:**
   ```
   n
   n
   y
   ```

4. Given a BST, print out all root-to-leaf paths.

5. Find the number of leaves in a BST.

6. Find sum of all the leaf nodes in a BST.

7. Construct a binary tree given inorder and post-order traversal outputs.

8. Construct a full binary tree from given pre-order and post-order traversals and print in-order traversal of it.

9. Delete a BST. Print the order in which nodes are deleted.

10. Construct the mirror tree of a given BST.

11. Given a Binary Search Tree, and an integer k. Print all the nodes which are at k distance from root.

12. Find out the in-order successor and predecessor of a given node in a BST.

13. Given a BST and a key, write a function that prints all the ancestors of the key in the given binary tree.

14. Write a function which deletes all the terminal nodes in BST.

15. Given a BST, delete all the nodes by repeated deletion of root. Print the inorder traversal after every deletion.

16. Given a binary tree, find if it is a BST.

17. A SumTree is a Binary Tree where the value of a node is equal to sum of the nodes present in its left subtree and right subtree. Write a function that returns 1 if the given BST is SumTree and 0 otherwise. All leaf nodes are trivial SumTrees.

18. Find distance between two given keys of a BST. Distance between two nodes is the minimum number of edges to be traversed to reach one node from other.

19. Write a function to print all the nodes in a BST along with their individual heights and depths.

20. Print output of depth-first search given a BST.

21. Print output of breadth-first search given a BST.

22. Remove all nodes which don't lie in any path with sum>= k. A node can be a part of multiple paths. So we have to delete it only in case when all paths from it have sum less than k. Print the in-order traversal of the tree after truncation.

23. Delete all duplicates of a node from a given binary search tree.