**k-th smallest element in BST**

**Medium**

Given a BST and an integer K. Find the Kth Smallest element in the BST using O(1) extra space.

**Example 1:**

**Input:**

2

  / \

  1 3

K = 2

**Output:** 2

**Explanation:** 2 is the 2nd smallest element in the BST

**Example 2:**

**Input:**

2

  / \

  1 3

K = 5

**Output:** -1

**Explanation:** There is no 5th smallest element in the BST as the size of BST is 3

**Expected Time Complexity:**O(N).  
**Expected Auxiliary Space:**O(1).  
  
**Constraints:**  
1<=Number of nodes<=10^5

**Company Tags**

[**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Google**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Google)

//{ Driver Code Starts

// Initial Template for Java

import java.io.\*;

import java.util.\*;

import java.math.\*;

class Node {

int data;

Node left, right;

public Node(int d) {

data = d;

left = right = null;

}

}

class CodingMaxima {

static Node buildTree(String str) {

// Corner Case

if (str.length() == 0 || str.equals('N')) return null;

String[] s = str.split(" ");

Node root = new Node(Integer.parseInt(s[0]));

Queue<Node> q = new LinkedList<Node>();

q.add(root);

// Starting from the second element

int i = 1;

while (!q.isEmpty() && i < s.length) {

// Get and remove the front of the queue

Node currrNode = q.remove();

// Get the currrent node's value from the string

String currrVal = s[i];

// If the left child is not null

if (!currrVal.equals("N")) {

// Create the left child for the currrent node

currrNode.left = new Node(Integer.parseInt(currrVal));

// Push it to the queue

q.add(currrNode.left);

}

// For the right child

i++;

if (i >= s.length) break;

currrVal = s[i];

// If the right child is not null

if (!currrVal.equals("N")) {

// Create the right child for the currrent node

currrNode.right = new Node(Integer.parseInt(currrVal));

// Push it to the queue

q.add(currrNode.right);

}

i++;

}

return root;

}

public static void main(String args[]) throws IOException {

BufferedReader br =

new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine().trim());

while (t > 0) {

String s = br.readLine();

Node root = buildTree(s);

int k = Integer.parseInt(br.readLine().trim());

Solution T = new Solution();

System.out.println(T.KthSmallestElement(root, k));

t--;

}

}

}

// } Driver Code Ends

class Solution {

private int c=0;

private int ans=-1;

public void Bst(Node root , int K){

if (root==null)

return ;

Bst(root.left, K);

c++;

if(c==K){

ans=root.data;

return ;

}

Bst(root.right,K);

}

public int KthSmallestElement(Node root, int K) {

Bst(root, K);

return ans;

}

}