**Problem Statement**

**Title: Find K-th Smallest Keyword in BST**

**Problem Description:** Lucas is managing a search engine and wants to find the k-th smallest keyword in the binary search tree (BST) of keywords. He needs to efficiently retrieve the k-th smallest element. Given the root of a binary search tree, and an integer k, print the k-th smallest value (1-indexed) of all the values of the nodes in the tree.

**Input Format:**

* The first line contains the values of all the nodes in the binary tree in pre-order format where true suggests the node exists and false suggests it is NULL.
* The second line contains the value of k.

**Output Format:**

Print the integer value of the k-th smallest keyword.

**Constraints:**

* The number of nodes in the tree is in the range [0, 2000].
* -1000 <= Node.val <= 1000

**Examples**

**Example 1:**

Input:

3 true 1 true 0 false false true 2 false false true 4 true 3 false false false

2

Output:

1

**Example 2:**

Input:

5 true 3 true 2 true 1 false false false true 4 false false true 6 false false

3

Output:

3

**Test Cases**

**Test Case 1:**

Input:

7 true 3 true 1 false false true 5 true 4 false false true 6 false false true 10 true 8 false false true 12 false false

5

Output:

6

**Test Case 2:**

Input:

10 true 5 true 3 true 2 false false false true 7 true 6 false false true 8 false false true 15 true 12 true 11 false false false true 20 false false

4

Output:

6

**Test Case 3:**

Input:

8 true 3 true 1 false false true 6 true 4 false false true 7 false false true 10 false true 14 true 13 false false true 15 false false

6

Output:

8

**Test Case 4:**

Input:

4 true 2 true 1 false false true 3 false false true 6 true 5 false false true 7 false false

3

Output:

3

**Test Case 5:**

Input:

9 true 5 true 3 true 1 false false false true 7 true 6 false false true 8 false false true 12 true 11 false false true 14 true 13 false false

7

Output:

9

**Solution in Python**

Here is a Python solution that constructs the binary tree from pre-order input, performs an in-order traversal to find the k-th smallest element, and prints it:

python

Copy code

class TreeNode:

def \_\_init\_\_(self, value=0, left=None, right=None):

self.value = value

self.left = left

self.right = right

def build\_tree(values):

if not values:

return None

def build\_tree\_helper(index):

if index >= len(values) or values[index] == 'false':

return None, index

node = TreeNode(int(values[index]))

index += 1

if index < len(values) and values[index] == 'true':

node.left, index = build\_tree\_helper(index + 1)

else:

index += 1

if index < len(values) and values[index] == 'true':

node.right, index = build\_tree\_helper(index + 1)

else:

index += 1

return node, index

root, \_ = build\_tree\_helper(0)

return root

def kth\_smallest(root, k):

def inorder\_traversal(node):

if not node:

return []

return inorder\_traversal(node.left) + [node.value] + inorder\_traversal(node.right)

inorder = inorder\_traversal(root)

return inorder[k-1] if k-1 < len(inorder) else None

def main():

import sys

input = sys.stdin.read

data = input().strip().split()

tree\_data = data[:-1]

k = int(data[-1])

root = build\_tree(tree\_data)

print(kth\_smallest(root, k))

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

import sys

from io import StringIO

sys.stdin = StringIO("3 true 1 true 0 false false true 2 false false true 4 true 3 false false false\n2")

main() # Output: 1

This solution reads the input values, constructs the binary search tree using the given pre-order format, and finds the k-th smallest element using an in-order traversal