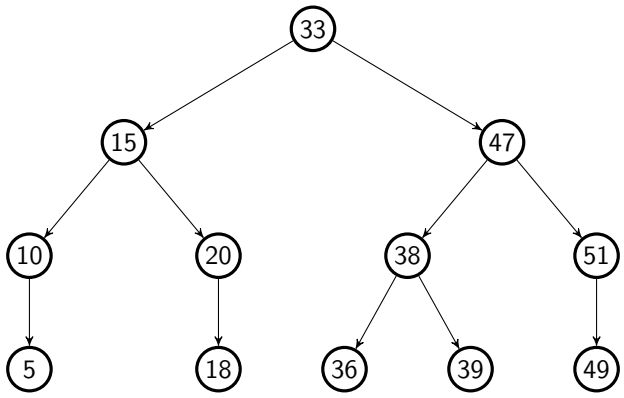
CS29003 Algorithms Laboratory
Level Order Traversal

General instruction to be followed strictly

1. Do not use any global variable unless you are explicitly instructed so.
2. Do not use Standard Template Library (STL) of C++.
3. Use proper indentation in your code and comment.
4. Name your file as `<roll_no>_<assignment_no>`. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c` or `14CS10001_3.cpp` as applicable.
5. Write your name, roll number, and assignment number at the beginning of your program.

In the class, we have studied preorder, inorder, and postorder traversals for binary trees. There is another traversal called *level order traversal* for binary trees. In the level order traversal, all the nodes with distance k from the root node are visited before any node with distance more than k from the root node for every $k \in \mathbb{N}$. Moreover, if the distances of any two nodes x and y from the root node are the same, then x is visited before y if and only if x appears on the left of y. Figure 1 shows a binary tree with its level order traversal.



Level order traversal: 33, 15, 47, 10, 20, 38, 51, 5, 18, 36, 39, 49

Figure 1: A binary tree and its level order traversal.

Use the following struct to define a node of a binary tree.

```
typedef struct node{
        int value;
        struct node *left;
        struct node *right;
}NODE, *NODEPRT;
```

Part 1: Construct a binary search tree from a level order traversal

In the class, we have learned how a unique binary search tree can be constructed if one of the preorder or postorder traversals are given. Use those ideas to write a program which takes a level order traversal as input and creates the unique binary search tree with that level order traversal. Use the following prototype for implementing your algorithm. You may use $\mathcal{O}(n)$ extra space.

NODEPTR createBST(int*, int);

Print the preorder and inorder traversal of the binary search tree created above. Use the following prototype for implementing the preorder, inorder, and postorder traversals.

void preOrder(NODEPTR);
void inOrder(NODEPTR);
void PostOrder(NODEPTR);

Part 2: Find k-th Minimum

Write a function which takes a binary search tree and an integer k as input and outputs the k-th minimum element of the binary search tree. If k is greater than the number of nodes in the tree, then display the number of nodes in the tree. Use the following prototype.

void min_k(NODEPTR, int);

Part 3: Change Root

In this part, take an integer ℓ as input from the user and check if ℓ is present in the binary search tree. If ℓ is not present, print "the key ℓ is not present." Otherwise, change the left and right pointers of the nodes of the binary search tree so that the node with key ℓ becomes the root node. Do not create any extra node or change key value of any node of the binary search tree. You are allowed to change left and right pointers of the nodes only. Use the following prototype.

NODEPTR find_and_change(NODEPTR,int);

Print the preorder, inorder, and postorder traversals of the modified binary search tree by calling the functions written in part 1.

Sample Output

Enter level order traversal for creating a binary search tree: 5, 3, 7, 4, 6, 9

Preorder traversal: 5, 3, 4, 7, 6, 9
Inorder traversal: 3, 4, 5, 6, 7, 9
Postorder traversal: 4, 3, 6, 9, 7, 5

Enter k: 3

3-th smallest element is 5

Enter ℓ: 3

Preorder traversal of the new binary tree: 7, 6, 5, 3, 4, 9
Inorder traversal of the new binary tree: 3, 4, 5, 6, 7, 9
Postorder traversal of the new binary tree: 4, 3, 5, 6, 9, 7