

# Rajalakshmi Engineering College

Name: Jothika k  
Email: 241801110@rajalakshmi.edu.in  
Roll no: 241801110  
Phone: 8015818072  
Branch: REC  
Department: I AI & DS FB  
Batch: 2028  
Degree: B.E - AI & DS

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

John is learning about Binary Search Trees (BST) in his computer science class. He wants to create a program that allows users to delete a node with a given value from a BST and print the remaining nodes using an in-order traversal.

Implement a function to help him delete a node with a given value from a BST.

##### ***Input Format***

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the BST nodes.

The third line consists of an integer V, which is the value to delete from the BST.

### **Output Format**

The output prints the space-separated values in the BST in an in-order traversal, after the deletion of the specified value.

If the specified value is not available in the tree, print the given input values in-order traversal.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5  
10 5 15 2 7  
15

Output: 2 5 7 10

### **Answer**

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};
```

```
struct TreeNode* createNode(int key) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
    newNode->data = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}
```

```
struct TreeNode* insert(struct TreeNode* root, int key) {
    if (!root) return createNode(key);
```

```
    if (key < root->data) root->left = insert(root->left, key);  
    else root->right = insert(root->right, key);  
    return root;  
}
```

```
struct TreeNode* findMin(struct TreeNode* node) {  
    while (node->left) node = node->left;  
    return node;  
}
```

```
struct TreeNode* deleteNode(struct TreeNode* root, int key) {  
    if (!root) return root;  
  
    if (key < root->data)  
        root->left = deleteNode(root->left, key);  
    else if (key > root->data)  
        root->right = deleteNode(root->right, key);  
    else {  
        if (!root->left) {  
            TreeNode* temp = root->right;  
            free(root);  
            return temp;  
        } else if (!root->right) {  
            TreeNode* temp = root->left;  
            free(root);  
            return temp;  
        }  
  
        TreeNode* temp = findMin(root->right);  
        root->data = temp->data;  
        root->right = deleteNode(root->right, temp->data);  
    }  
    return root;  
}
```

```
void inorderTraversal(struct TreeNode* root) {  
    if (root) {  
        inorderTraversal(root->left);
```

```
        printf("%d ", root->data);  
        inorderTraversal(root->right);  
    }  
}
```

```
int main()  
{  
    int N, rootValue, V;  
    scanf("%d", &N);  
    struct TreeNode* root = NULL;  
    for (int i = 0; i < N; i++) {  
        int key;  
        scanf("%d", &key);  
        if (i == 0) rootValue = key;  
        root = insert(root, key);  
    }  
    scanf("%d", &V);  
    root = deleteNode(root, V);  
    inorderTraversal(root);  
    return 0;  
}
```

**Status :** Correct

**Marks :** 10/10