

# Rajalakshmi Engineering College

Name: krithika narasimhan  
Email: 240701277@rajalakshmi.edu.in  
Roll no: 240701277  
Phone: 9677451731  
Branch: REC  
Department: I CSE FC  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_CY\_Updated

Attempt : 1  
Total Mark : 30  
Marks Obtained : 30

### Section 1 : Coding

#### 1. Problem Statement

Kishore is studying data structures, and he is currently working on implementing a binary search tree (BST) and exploring its basic operations. He wants to practice creating a BST, inserting elements into it, and performing a specific operation, which is deleting the minimum element from the tree.

Write a program to help him perform the delete operation.

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

The first line of input consists of an integer N, representing the number of elements Kishore wants to insert into the BST.

The second line consists of N space-separated integers, where each integer represents an element to be inserted into the BST.

### **Output Format**

The output prints the remaining elements of the BST in ascending order (in-order traversal) after deleting the minimum element.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 6

5 3 8 2 4 6

Output: 3 4 5 6 8

### **Answer**

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

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

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

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

```
struct Node* insert(struct Node* root, int data) {  
    if (root == NULL)  
        return newNode(data);  
  
    if (data < root->data)  
        root->left = insert(root->left, data);  
    else if (data > root->data)
```

```
    root->right = insert(root->right, data);  
    return root;  
}
```

```
struct Node* minValueNode(struct Node* node) {  
    struct Node* current = node;  
    while (current && current->left != NULL)  
        current = current->left;  
    return current;  
}
```

```
struct Node* deleteMin(struct Node* root) {  
    if (root == NULL)  
        return root;  
  
    if (root->left == NULL) {  
        struct Node* temp = root->right;  
        free(root);  
        return temp;  
    }  
  
    root->left = deleteMin(root->left);  
    return root;  
}
```

```
void inOrderTraversal(struct Node* root) {  
    if (root) {  
        inOrderTraversal(root->left);  
        printf("%d ", root->data);  
        inOrderTraversal(root->right);  
    }  
}
```

```
int main() {  
    struct Node* root = NULL;  
    int n, data;  
  
    scanf("%d", &n);  
  
    for (int i = 0; i < n; i++) {  
        scanf("%d", &data);
```

```
    root = insert(root, data);  
}  
  
root = deleteMin(root);  
  
inOrderTraversal(root);  
return 0;  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

John is building a system to store and manage integers using a binary search tree (BST). He needs to add a feature that allows users to search for a specific integer key in the BST using recursion.

Implement functions to create the BST and perform a recursive search for an integer.

### **Input Format**

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

The second line consists of integers representing, the values of nodes, separated by space.

The third line consists of an integer representing, the key to be searched.

### **Output Format**

The output prints whether the given key is present in the binary search tree or not.

Refer to the sample output for the exact format.

### **Sample Test Case**

Input: 7  
10 5 15 3 7 12 20

12

Output: The key 12 is found in the binary search tree

**Answer**

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* left;
    Node* right;
};

Node* createNode(int value) {
    Node* newNode = new Node();
    if (!newNode) {
        cout << "Memory allocation error!" << endl;
        return NULL;
    }
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

Node* insertNode(Node* root, int value) {
    if (root == NULL) {
        return createNode(value);
    }

    if (value < root->data) {
        root->left = insertNode(root->left, value);
    } else if (value > root->data) {
        root->right = insertNode(root->right, value);
    }

    return root;
}

bool searchKey(Node* root, int key) {
    if (root == NULL || root->data == key) {
        return root != NULL;
    }
}
```

```

    if (key < root->data) {
        return searchKey(root->left, key);
    } else {
        return searchKey(root->right, key);
    }
}

int main() {
    Node* root = NULL;
    int numNodes, value, key;

    cin >> numNodes;

    for (int i = 0; i < numNodes; i++) {
        cin >> value;
        root = insertNode(root, value);
    }

    cin >> key;

    bool found = searchKey(root, key);
    if (found) {
        cout << "The key " << key << " is found in the binary search tree" << endl;
    } else {
        cout << "The key " << key << " is not found in the binary search tree" << endl;
    }

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Dhruv is working on a project where he needs to implement a Binary Search Tree (BST) data structure and perform various operations on it.

He wants to create a program that allows him to build a BST, traverse it in different orders (inorder, preorder, postorder), and exit the program when

needed.

Help Dhruv by designing a program that fulfils his requirements.

### ***Input Format***

The first input consists of the choice.

If the choice is 1, enter the number of elements N and the elements inserted into the tree, separated by a space in a new line.

If the choice is 2, print the in-order traversal.

If the choice is 3, print the pre-order traversal.

If the choice is 4, print the post-order traversal.

If the choice is 5, exit.

### ***Output Format***

The output prints the results based on the choice.

For choice 1, print "BST with N nodes is ready to use" where N is the number of nodes inserted.

For choice 2, print the in-order traversal of the BST.

For choice 3, print the pre-order traversal of the BST.

For choice 4, print the post-order traversal of the BST.

For choice 5, the program exits.

If the choice is greater than 5, print "Wrong choice".

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 1

5  
12 78 96 34 55  
2  
3  
4  
5

Output: BST with 5 nodes is ready to use

BST Traversal in INORDER

12 34 55 78 96

BST Traversal in PREORDER

12 78 34 55 96

BST Traversal in POSTORDER

55 34 96 78 12

### **Answer**

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

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

```
struct tnode
```

```
{
```

```
    int data;
```

```
    struct tnode *right;
```

```
    struct tnode *left;
```

```
};
```

```
struct tnode *CreateBST(struct tnode *root, int item)
```

```
{
```

```
    if(root == NULL)
```

```
    {
```

```
        root = (struct tnode *)malloc(sizeof(struct tnode));
```

```
        root->left = root->right = NULL;
```

```
        root->data = item;
```

```
        return root;
```

```
    }
```

```
    else
```

```
    {
```

```
        if(item < root->data )
```

```
            root->left = CreateBST(root->left,item);
```

```
        else if(item > root->data )
```

```
            root->right = CreateBST(root->right,item);
```

```
        return(root);
```

```
    }
```



```
}
```

```
void Inorder(struct tnode *root)
```

```
{
```

```
    if( root != NULL)
```

```
    {
```

```
        Inorder(root->left);
```

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

```
        Inorder(root->right);
```

```
    }
```

```
}
```

```
void Preorder(struct tnode *root)
```

```
{
```

```
    if( root != NULL)
```

```
    {
```

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

```
        Preorder(root->left);
```

```
        Preorder(root->right);
```

```
    }
```

```
}
```

```
void Postorder(struct tnode *root)
```

```
{
```

```
    if( root != NULL)
```

```
    {
```

```
        Postorder(root->left);
```

```
        Postorder(root->right);
```

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

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    struct tnode *root = NULL;
```

```
    int choice, item, n, i;
```

```
    do
```

```
    {
```

```
        scanf("%d",&choice);
```

```
        switch(choice)
```

```
        {
```

```
            case 1:
```

```
                root = NULL;
```

```

scanf("%d",&n);
for(i = 1; i <= n; i++)
{
    scanf("%d",&item);
    root = CreateBST(root,item);
}
printf("BST with %d nodes is ready to use\n", n);
break;
case 2:
    printf("BST Traversal in INORDER\n");
    Inorder(root);
    printf("\n");
    break;
case 3:
    printf("BST Traversal in PREORDER\n");
    Preorder(root);
    printf("\n");
    break;
case 4:
    printf("BST Traversal in POSTORDER\n");
    Postorder(root);
    printf("\n");
    break;
case 5:
    exit(0);
    break;
default:
    printf("Wrong choice\n");
    break;
}
} while(1);
return 0;
}

```

**Status :** Correct

**Marks :** 10/10