

# Rajalakshmi Engineering College

Name: Harish Suresh  
Email: 241901034@rajalakshmi.edu.in  
Roll no: 241901034  
Phone: 7550029270  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 15

#### Section 1 : MCQ

1. Which of the following is the correct in-order traversal of a binary search tree with nodes: 9, 3, 5, 11, 8, 4, 2?

**Answer**

2, 3, 4, 5, 8, 9, 11

**Status : Correct**

**Marks : 1/1**

2. Find the postorder traversal of the given binary search tree.

**Answer**

1, 4, 2, 18, 14, 13

**Status : Correct**

**Marks : 1/1**

3. While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is \_\_\_\_\_.

**Answer**

67

**Status :** Correct

**Marks :** 1/1

4. The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19. Which one of the following is the postorder traversal of the tree?

**Answer**

11, 12, 10, 16, 19, 18, 20, 15

**Status :** Correct

**Marks :** 1/1

5. Which of the following is a valid preorder traversal of the binary search tree with nodes: 18, 28, 12, 11, 16, 14, 17?

**Answer**

18, 12, 11, 16, 14, 17, 28

**Status :** Correct

**Marks :** 1/1

6. How many distinct binary search trees can be created out of 4 distinct keys?

**Answer**

14

**Status :** Correct

**Marks :** 1/1

7. Find the post-order traversal of the given binary search tree.

**Answer**

10, 17, 20, 18, 15, 32, 21

**Status :** Correct

**Marks :** 1/1

8. Find the pre-order traversal of the given binary search tree.

**Answer**

13, 2, 1, 4, 14, 18

**Status :** Correct

**Marks :** 1/1

9. Which of the following is the correct post-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

**Answer**

20, 32, 30, 52, 57, 55, 50

**Status :** Correct

**Marks :** 1/1

10. In a binary search tree with nodes 18, 28, 12, 11, 16, 14, 17, what is the value of the left child of the node 16?

**Answer**

14

**Status :** Correct

**Marks :** 1/1

11. Find the in-order traversal of the given binary search tree.

**Answer**

1, 2, 4, 13, 14, 18

**Status :** Correct

**Marks :** 1/1

12. Which of the following is the correct pre-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

**Answer**

50, 30, 20, 32, 55, 52, 57

**Status :** Correct

**Marks :** 1/1

13. While inserting the elements 5, 4, 2, 8, 7, 10, 12 in a binary search tree, the element at the lowest level is \_\_\_\_\_.

**Answer**

12

**Status :** Correct

**Marks :** 1/1

14. Find the preorder traversal of the given binary search tree.

**Answer**

9, 2, 1, 6, 4, 7, 10, 14

**Status :** Correct

**Marks :** 1/1

15. Which of the following operations can be used to traverse a Binary Search Tree (BST) in ascending order?

**Answer**

Inorder traversal

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Harish Suresh  
Email: 241901034@rajalakshmi.edu.in  
Roll no: 241901034  
Phone: 7550029270  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

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 == NULL) return createNode(key);
    if (key < root->data)
```

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

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

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



# Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

Batch: 2028

Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

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

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

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

#### ***Output Format***

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

3 1 5 2 4

Output: 3 1 2 5 4

### **Answer**

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

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

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

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

```
void printPreorder(struct Node* node) {  
    if (node == NULL)  
        return;  
    printf("%d ", node->data);  
    printPreorder(node->left);  
    printPreorder(node->right);  
}
```

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

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Harish Suresh  
Email: 241901034@rajalakshmi.edu.in  
Roll no: 241901034  
Phone: 7550029270  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

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

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

in the binary search tree.

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

The third line consists of an integer representing, the value that is to be searched.

### **Output Format**

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 7

8 3 10 1 6 14 23

6

Output: Value 6 is found in the tree.

### **Answer**

```
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;
}

Node* searchNode(Node* root, int value) {
    if (root == NULL || root->data == value) {
        return root;
    }
    if (value < root->data) {
        return searchNode(root->left, value);
    }
    return searchNode(root->right, value);
}
```

}

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Harish Suresh  
Email: 241901034@rajalakshmi.edu.in  
Roll no: 241901034  
Phone: 7550029270  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 4

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing the program.

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

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

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

### **Output Format**

The first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3

5 10 15

Output: 15 10 5

The minimum value in the BST is: 5

### **Answer**

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

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

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

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

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



```

    }
    return root;
}

void displayTreePostOrder(struct Node* root) {
    if (root == NULL) {
        return;
    }
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ", root->data);
}

int findMinValue(struct Node* root) {
    if (root == NULL) {
        return 1000000;
    }
    int leftMin = findMinValue(root->left);
    int rightMin = findMinValue(root->right);
    int min = root->data;
    if (leftMin < min) {
        min = leftMin;
    }
    if (rightMin < min) {
        min = rightMin;
    }
    return min;
}

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);
    }

    displayTreePostOrder(root);
    printf("\n");

    int minValue = findMinValue(root);
    printf("The minimum value in the BST is: %d", minValue);
}

```

```
} return 0;
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

Batch: 2028

Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

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

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

#### ***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 nodes to insert into the BST.

#### ***Output Format***

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

10 5 15 2 7

Output: 15

### **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 == NULL) return createNode(key);
    if (key < root->data)
        root->left = insert(root->left, key);
    else if (key > root->data)
        root->right = insert(root->right, key);
    return root;
}

int findMax(struct TreeNode* root) {
    if (root == NULL) return -1;
    while (root->right != NULL) {
        root = root->right;
    }
}
```

```
        return root->data;
    }
    int main() {
        int N, rootValue;
        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);
        }

        int maxVal = findMax(root);
        if (maxVal != -1) {
            printf("%d", maxVal);
        }

        return 0;
    }
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Harish Suresh  
Email: 241901034@rajalakshmi.edu.in  
Roll no: 241901034  
Phone: 7550029270  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

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

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

#### Section 1 : Coding

##### 1. Problem Statement

Yogi is working on a program to manage a binary search tree (BST) containing integer values. He wants to implement a function that removes nodes from the tree that fall outside a specified range defined by a minimum and maximum value.

Help Yogi by writing a function that achieves this.

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

The first line of input consists of an integer N, representing the number of elements to be inserted into the BST.

The second line consists of N space-separated integers, representing the elements to be inserted into the BST.

The third line consists of two space-separated integers min and max, representing the minimum value and the maximum value of the range.

### **Output Format**

The output prints the remaining elements of the BST in an in-order traversal, after removing nodes that fall outside the specified range.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

10 5 15 20 12

5 15

Output: 5 10 12 15

### **Answer**

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

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

```
struct node {  
    int key;  
    struct node* left;  
    struct node* right;  
};
```

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

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

```

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

struct node* removeOutsideRange(struct node* root, int min, int max) {
    if (root == NULL)
        return NULL;

    root->left = removeOutsideRange(root->left, min, max);
    root->right = removeOutsideRange(root->right, min, max);

    if (root->key < min) {
        struct node* rChild = root->right;
        free(root);
        return rChild;
    }

    if (root->key > max) {
        struct node* lChild = root->left;
        free(root);
        return lChild;
    }

    return root;
}

void inorderTraversal(struct node* root) {
    if (root) {
        inorderTraversal(root->left);
        printf("%d ", root->key);
        inorderTraversal(root->right);
    }
}

int main() {
    struct node* root = NULL;
    int num, min, max;

    scanf("%d", &num);

    for (int i = 0; i < num; i++) {

```



```
int key;  
scanf("%d", &key);  
root = insert(root, key);  
}  
  
scanf("%d", &min);  
scanf("%d", &max);  
  
root = removeOutsideRange(root, min, max);  
  
inorderTraversal(root);  
  
return 0;  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Aishu is participating in a coding challenge where she needs to reconstruct a Binary Search Tree (BST) from given preorder traversal data and then print the in-order traversal of the reconstructed BST.

Since Aishu is just learning about tree data structures, she needs your help to write a program that does this efficiently.

### **Input Format**

The first line consists of an integer  $n$ , representing the number of nodes in the BST.

The second line of input contains  $n$  integers separated by spaces, which represent the preorder traversal of the BST.

### **Output Format**

The output displays  $n$  space-separated integers, representing the in-order traversal of the reconstructed BST.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 6

10 5 1 7 40 50

Output: 1 5 7 10 40 50

### **Answer**

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

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

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

```
struct node* newNode(int data) {  
    struct node* temp = (struct node*)malloc(sizeof(struct node));  
    if (!temp) {  
        printf("Memory allocation error!\n");  
        return NULL;  
    }  
    temp->data = data;  
    temp->left = temp->right = NULL;  
    return temp;  
}
```

```
struct node* constructTreeUtil(int pre[], int* preIndex, int low, int high, int size) {  
    if (*preIndex >= size || low > high)  
        return NULL;
```

```
    struct node* root = newNode(pre[*preIndex]);  
    *preIndex = *preIndex + 1;
```

```
    if (low == high)  
        return root;
```

```
    int i;  
    for (i = low; i <= high; ++i)  
        if (pre[i] > root->data)  
            break;
```

```

    root->left = constructTreeUtil(pre, preIndex, *preIndex, i - 1, size);
    root->right = constructTreeUtil(pre, preIndex, i, high, size);

    return root;
}

struct node* constructTree(int pre[], int size) {
    int preIndex = 0;
    return constructTreeUtil(pre, &preIndex, 0, size - 1, size);
}

void printInorder(struct node* node) {
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}

int main() {
    int i, size;
    scanf("%d", &size);
    int* pre = (int*)malloc(size * sizeof(int));
    if (!pre) {
        printf("Memory allocation error!\n");
        return 1;
    }

    for (i = 0; i < size; i++) {
        scanf("%d", &pre[i]);
    }

    struct node* root = constructTree(pre, size);
    printInorder(root);

    free(pre);
    return 0;
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Arun is exploring operations on binary search trees (BST). He wants to write a program with an unsorted distinct integer array that represents the BST keys and construct a height-balanced BST from it.

After constructing, he wants to perform the following operations that can alter the structure of the tree and traverse them using a level-order traversal:

InsertionDeletion

Your task is to assist Arun in completing the program without any errors.

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

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

The second line consists of N space-separated integers, representing the initial keys.

The third line consists of an integer X, representing the new key to be inserted into the BST.

The fourth line consists of an integer Y, representing the key to be deleted from the BST.

#### ***Output Format***

The first line of output prints "Initial BST: " followed by a space-separated list of keys in the initial BST after constructing it in level order traversal.

The second line prints "BST after inserting a new node X: " followed by a space-separated list of keys in the BST after inserting X n level order traversal.

The third line prints "BST after deleting node Y: " followed by a space-separated list of keys in the BST after deleting Y n level order traversal.

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 5

25 14 56 28 12

34

12

Output: Initial BST: 25 14 56 12 28

BST after inserting a new node 34: 25 14 56 12 28 34

BST after deleting node 12: 25 14 56 28 34

### Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
struct TreeNode {
    int val;
    struct TreeNode *left, *right;
};
struct TreeNode* createNode(int x) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
    newNode->val = x;
    newNode->left = newNode->right = NULL;
    return newNode;
}
```

```
struct TreeNode* insert(struct TreeNode* root, int val) {
    if (!root) return createNode(val);
    if (val < root->val) root->left = insert(root->left, val);
    else root->right = insert(root->right, val);
    return root;
}
```

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

```

        struct TreeNode* temp = root->left;
        free(root);
        return temp;
    }
    struct TreeNode* temp = root->right;
    while (temp && temp->left) temp = temp->left;
    root->val = temp->val;
    root->right = deleteNode(root->right, temp->val);
}
return root;
}

void levelOrderTraversal(struct TreeNode* root) {
    if (!root) return;
    struct TreeNode* queue[1000];
    int front = 0, rear = 0;
    queue[rear++] = root;
    while (front < rear) {
        struct TreeNode* current = queue[front++];
        printf("%d ", current->val);
        if (current->left) queue[rear++] = current->left;
        if (current->right) queue[rear++] = current->right;
    }
}

int main() {
    int n, x, y;
    scanf("%d", &n);
    struct TreeNode* root = NULL;
    for (int i = 0; i < n; ++i) {
        int val;
        scanf("%d", &val);
        root = insert(root, val);
    }

    printf("Initial BST: ");
    levelOrderTraversal(root);
    printf("\n");
    scanf("%d", &x);
    root = insert(root, x);
    printf("BST after inserting a new node %d: ", x);
    levelOrderTraversal(root);
    printf("\n");
}

```

```
scanf("%d", &y);
root = deleteNode(root, y);
printf("BST after deleting node %d: ", y);
levelOrderTraversal(root);
printf("\n");

return 0;
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Joseph, a computer science student, is interested in understanding binary search trees (BST) and their node arrangements. He wants to create a program to explore BSTs by inserting elements into a tree and displaying the nodes using post-order traversal of the tree.

Write a program to help Joseph implement the program.

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

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

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

##### ***Output Format***

The output prints N space-separated integer values after the post-order traversal.

Refer to the sample output for formatting specifications.

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

Input: 4  
10 15 5 3

Output: 3 5 15 10

**Answer**

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

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

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

struct Node* insert(struct Node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

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

    return root;
}

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



```
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);
    }

    displayTree(root);

    return 0;
}
```

**Status :** Correct

**Marks :** 10/10

## 5. Problem Statement

Viha, a software developer, is working on a project to automate searching for a target value in a Binary Search Tree (BST). She needs to create a program that takes an integer target value as input and determines if that value is present in the BST or not.

Write a program to assist Viha.

### **Input Format**

The first line of input consists of integers separated by spaces, which represent the elements to be inserted into the BST. The input is terminated by entering -1.

The second line consists of an integer target, which represents the target value to be searched in the BST.

### **Output Format**

If the target value is found in the BST, print "[target] is found in the BST".

Else, print "[target] is not found in the BST"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5 3 7 1 4 6 8 -1

4

Output: 4 is found in the BST

### **Answer**

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

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

```
#include <stdbool.h>
```

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

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

```
struct Node* insert(struct Node* root, int data) {  
    if (root == NULL) {  
        return createNode(data);  
    }
```

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

```
    return root;  
}
```

```
bool search(struct Node* root, int target) {  
    if (root == NULL) {
```

```

        return false;
    }

    if (root->data == target) {
        return true;
    }

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

int main() {
    struct Node* root = NULL;

    int num;
    while (1) {
        scanf("%d", &num);
        if (num == -1) {
            break;
        }
        root = insert(root, num);
    }

    int target;
    scanf("%d", &target);

    if (search(root, target)) {
        printf("%d is found in the BST", target);
    } else {
        printf("%d is not found in the BST", target);
    }

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

Batch: 2028

Degree: B.E - CSE (CS)

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

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 <stdio.h>
```

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

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

```
struct Node* createNode(int value) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    if (!newNode) {  
        printf("Memory allocation error!\n");  
        return NULL;  
    }  
    newNode->data = value;  
    newNode->left = newNode->right = NULL;  
    return newNode;  
}
```

```
struct Node* insertNode(struct 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;
}

```

```

int searchKey(struct Node* root, int key) {
    if (root == NULL) {
        return 0; // Not found
    }

    if (root->data == key) {
        return 1; // Found
    }

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

```

```

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

    scanf("%d", &numNodes);

    for (int i = 0; i < numNodes; i++) {
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    scanf("%d", &key);

    if (searchKey(root, key)) {
        printf("The key %d is found in the binary search tree\n", key);
    } else {
        printf("The key %d is not found in the binary search tree\n", key);
    }
}

```

```
}  
return 0;  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Arun is working on a Binary Search Tree (BST) data structure. His goal is to implement a program that reads a series of integers and inserts them into a BST. Once the integers are inserted, he needs to add a given integer value to each node in the tree and find the maximum value in the BST.

Your task is to help Arun implement this program.

### **Input Format**

The first line of input consists of an integer N, representing the number of elements to be inserted into the BST.

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

The third line consists of an integer add, representing the value to be added to each node in the BST.

### **Output Format**

The output prints the maximum value in the BST after adding the add value.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5  
10 5 15 20 25

5

Output: 30

### Answer

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

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

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

```
struct Node* insert(struct Node* root, int value) {
    if (root == NULL) {
        return createNode(value);
    }
    if (value < root->value) {
        root->left = insert(root->left, value);
    } else {
        root->right = insert(root->right, value);
    }
    return root;
}
```

```
void addToAllNodes(struct Node* root, int addValue) {
    if (root != NULL) {
        root->value += addValue;
        addToAllNodes(root->left, addValue);
        addToAllNodes(root->right, addValue);
    }
}
```

```
int findMax(struct Node* root) {
    if (root == NULL) {
        return -1;
    }
    while (root->right != NULL) {
```



```

        root = root->right;
    }
    return root->value;
}

int main() {
    int N, addValue;

    scanf("%d", &N);

    struct Node* root = NULL;

    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        root = insert(root, value);
    }

    scanf("%d", &addValue);

    addToAllNodes(root, addValue);

    int maxValue = findMax(root);

    printf("%d\n", maxValue);

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Emily is studying binary search trees (BST). She wants to write a program that inserts characters into a BST and then finds and prints the minimum and maximum values.

Guide her with the program.

**Input Format**

The first line of input consists of an integer N, representing the number of values to be inserted into the BST.

The second line consists of N space-separated characters.

### ***Output Format***

The first line of output prints "Minimum value: " followed by the minimum value of the given inputs.

The second line prints "Maximum value: " followed by the maximum value of the given inputs.

Refer to the sample outputs for formatting specifications.

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

Input: 5

Z E W T Y

Output: Minimum value: E

Maximum value: Z

### ***Answer***

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

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

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

```
struct Node* insert(struct Node* root, char data) {  
    if (root == NULL) {  
        root = (struct Node*)malloc(sizeof(struct Node));  
        if (!root) {  
            printf("Memory allocation error!\n");  
            return NULL;  
        }  
        root->data = data;  
        root->left = root->right = NULL;
```

```

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

```

```

char findMin(struct Node* root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root->data;
}

```

```

char findMax(struct Node* root) {
    while (root->right != NULL) {
        root = root->right;
    }
    return root->data;
}

```

```

int main() {
    struct Node* root = NULL;
    int size;
    scanf("%d", &size);
    char c;

    while (size != 0) {
        scanf(" %c", &c);
        root = insert(root, c);
        size--;
    }

    printf("Minimum value: %c\n", findMin(root));
    printf("Maximum value: %c\n", findMax(root));

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10