Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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 : 14

Section 1 : MCQ

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

Answer

14

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

Answer

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

Status : Correct   *Marks : 1/1*

4.   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*

5.   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*

6.   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*

7.   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*

8.   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*

9.   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*

10.   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*

11.   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*

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

Answer

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

Status : Correct   *Marks : 1/1*

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

Answer

1, 2, 4, 13, 14, 18

Status : Correct   *Marks : 1/1*

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

Answer

13, 2, 1, 4, 14, 18

Status : Correct   *Marks : 1/1*

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

Answer

4

Status : Wrong   *Marks : 0/1*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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

```c
#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;
}


// Function to insert a node into BST
```

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

// Function to find the minimum value node
struct TreeNode* findMin(struct TreeNode* node) {
    while (node->left != NULL) {
        node = node->left;
    }
    return node;
}

// Function to delete a node from BST
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);
```

```c
        }
        return root;
    }

    // Function to perform in-order traversal
    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: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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

```c
#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;
```

```c
    }

void printPreorder(struct Node* root) {
    if (root == NULL) {
        return;
    }

    printf("%d ", root->data);
    printPreorder(root->left);
    printPreorder(root->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: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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

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

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

// Function to create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a node into the BST
struct Node* insert(struct Node* root, int value) {
```

```c
    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;
}

// Function to search for a value in the BST
int search(struct Node* root, int key) {
    if (root == NULL) {
        return 0; // Not found
    }

    if (root->data == key) {
        return 1; // Found
    } else if (key < root->data) {
        return search(root->left, key);
    } else {
        return search(root->right, key);
    }
}

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

    // Read the number of nodes
    scanf("%d", &n);

    // Insert nodes into the BST
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }

    // Read the key to search
```

```c
    scanf("%d", &key);

    // Search for the key and print the result
    if (search(root, key)) {
        printf("Value %d is found in the tree.\n", key);
    } else {
        printf("Value %d is not found in the tree.\n", key);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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

```c
#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;
}

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

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

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

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;
}

void postOrderTraversal(struct TreeNode* root) {
    if (!root) return;
    postOrderTraversal(root->left);
    postOrderTraversal(root->right);
    printf("%d ", root->data);
}

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

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

    postOrderTraversal(root);
    printf("\nThe minimum value in the BST is: %d\n", findMin(root)->data);

    return 0;
}
int main() {
```

```c
    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: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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

```c
#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;
}
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

// Function to create a new node
struct TreeNode* createNode(int key) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
```

```c
    newNode->data = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a node into BST
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 {
        root->right = insert(root->right, key);
    }
    return root;
}

// Function to find the maximum value node in BST
struct TreeNode* findMax(struct TreeNode* root) {
    while (root->right != NULL) {
        root = root->right;
    }
    return root;
}

int main() {
    int N;
    scanf("%d", &N);

    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        root = insert(root, key);
    }

    // Find and print maximum value
    struct TreeNode* maxNode = findMax(root);
    printf("%d\n", maxNode->data);

    return 0;
```

```c
    }
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: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

struct TreeNode {
    int data;
```

```c
    struct TreeNode *left, *right;
};

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

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

void levelOrderTraversal(struct TreeNode* root) {
    if (!root) return;
    struct TreeNode* queue[100];
    int front = 0, rear = 0;
    queue[rear++] = root;
    while (front < rear) {
        struct TreeNode* temp = queue[front++];
        printf("%d ", temp->data);
        if (temp->left) queue[rear++] = temp->left;
        if (temp->right) queue[rear++] = temp->right;
```

```c
        }
    }

    int main() {
        int N, X, Y;
        scanf("%d", &N);
        struct TreeNode* root = NULL;
        for (int i = 0; i < N; i++) {
            int key;
            scanf("%d", &key);
            root = insert(root, key);
        }

        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*

2. 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

```c
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

struct TreeNode* createNode(int key) {
    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    node->data = key;
```

```c
        node->left = node->right = NULL;
        return node;
}

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* trimBST(struct TreeNode* root, int min, int max) {
    if (!root) return NULL;
    root->left = trimBST(root->left, min, max);
    root->right = trimBST(root->right, min, max);
    if (root->data < min) return root->right;
    if (root->data > max) return root->left;
    return root;
}

void inorderTraversal(struct TreeNode* root) {
    if (!root) return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorderTraversal(root->right);
}

int main() {
    int N, min, max;
    scanf("%d", &N);
    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        root = insert(root, key);
    }

    scanf("%d %d", &min, &max);
    root = trimBST(root, min, max);
    inorderTraversal(root);
    printf("\n");
```

```
    return 0;
}
```

3. 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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
```

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

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

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;
}

void postOrderTraversal(struct TreeNode* root) {
    if (!root) return;
    postOrderTraversal(root->left);
    postOrderTraversal(root->right);
    printf("%d ", root->data);
}

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

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

    return 0;
}
```

4.   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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

struct TreeNode {
    int data;
```

```c
    struct TreeNode *left, *right;
};

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

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* constructBST(int preorder[], int n) {
    struct TreeNode* root = NULL;
    for (int i = 0; i < n; i++) {
        root = insert(root, preorder[i]);
    }
    return root;
}

void inorderTraversal(struct TreeNode* root) {
    if (!root) return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorderTraversal(root->right);
}

int main() {
    int N;
    scanf("%d", &N);
    int preorder[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &preorder[i]);
    }

    struct TreeNode* root = constructBST(preorder, N);
    inorderTraversal(root);
```

```
    printf("\n");

    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

```c
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

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;
}

int searchBST(struct TreeNode* root, int target) {
    if (!root) return 0;
    if (root->data == target) return 1;
    return target < root->data ? searchBST(root->left, target) : searchBST(root-
>right, target);
}

int main() {
    struct TreeNode* root = NULL;
    int key, target;

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

    scanf("%d", &target);
    if (searchBST(root, target))
```

```c
        printf("%d is found in the BST\n", target);
    else
        printf("%d is not found in the BST\n", target);

    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: l 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 : 23.5

Section 1 : Coding

1. 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

```c
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

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;
}

void inorderTraversal(struct TreeNode* root) {
   if (!root) return;
   inorderTraversal(root->left);
   printf("%d ", root->data);
   inorderTraversal(root->right);
}

void preorderTraversal(struct TreeNode* root) {
   if (!root) return;
   printf("%d ", root->data);
   preorderTraversal(root->left);
   preorderTraversal(root->right);
}
```

```c
void postorderTraversal(struct TreeNode* root) {
    if (!root) return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ", root->data);
}

int main() {
    struct TreeNode* root = NULL;
    int choice, N, key;

    while (1) {
        scanf("%d", &choice);
        if (choice == 1) {
            scanf("%d", &N);
            for (int i = 0; i < N; i++) {
                scanf("%d", &key);
                root = insert(root, key);
            }
            printf("BST with %d nodes is ready to use\n", N);
        } else if (choice == 2) {
            printf("BST Traversal in INORDER\n");
            inorderTraversal(root);
            printf("\n");
        } else if (choice == 3) {
            printf("BST Traversal in PREORDER\n");
            preorderTraversal(root);
            printf("\n");
        } else if (choice == 4) {
            printf("BST Traversal in POSTORDER\n");
            postorderTraversal(root);
            printf("\n");
        } else if (choice == 5) {
            break;
        } else {
            printf("Wrong choice\n");
        }
    }

    return 0;
}
```

2. Problem Statement

Edward has a Binary Search Tree (BST) and needs to find the k-th largest element in it.

Given the root of the BST and an integer k, help Edward determine the k-th largest element in the tree. If k exceeds the number of nodes in the BST, return an appropriate message.

Input Format

The first line of input consists of integer n, the number of nodes in the BST.

The second line consists of the n elements, separated by space.

The third line consists of the value of k.

Output Format

The output prints the kth largest element in the binary search tree.

For invalid inputs, print "Invalid value of k".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 4 12 2 6 10 14
1
Output: 14

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
```

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

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

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;
}

void reverseInorder(struct TreeNode* root, int *count, int k, int *result) {
    if (!root || *count >= k) return;
    reverseInorder(root->right, count, k, result);
    (*count)++;
    if (*count == k) {
        *result = root->data;
        return;
    }
    reverseInorder(root->left, count, k, result);
}

int findKthLargest(struct TreeNode* root, int k) {
    int count = 0, result = -1;
    reverseInorder(root, &count, k, &result);
    return result;
}

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

```
    scanf("%d", &key);
    root = insert(root, key);
}

scanf("%d", &k);
int result = findKthLargest(root, k);
if (result == -1) printf("Invalid value of k\n");
else printf("%d\n", result);

return 0;
}
```

Status : Correct   *Marks : 10/10*

3.   Problem Statement

Jake is learning about binary search trees(BST) and their operations. He
wants to implement a program that can delete a node from a BST based
on the given key value and print the remaining nodes in an in-order
traversal.

Assist Jake in the program.

Input Format

The first line of input consists of an integer n, representing the number of
elements in BST.

The second line consists of n space-separated integers, representing the
elements of the tree.

The third line consists of an integer x, representing the key value of the node to
be deleted.

Output Format

The first line of output prints "Before deletion: " followed by the in-order traversal
of the initial BST.

The second line prints "After deletion: " followed by the in-order traversal after
the deletion of the key value.

If the key value is not present in the BST, print the original tree as it is.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 6 4 3 1
4
Output: Before deletion: 1 3 4 6 8
After deletion: 1 3 6 8

Answer

```c
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

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;
}

void inorderTraversal(struct TreeNode* root) {
    if (!root) return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
```

```c
        inorderTraversal(root->right);
}

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

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) {
            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 = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

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

    printf("Before deletion: ");
    inorderTraversal(root);
    printf("\n");
```

```c
        scanf("%d", &x);
        struct TreeNode* temp = deleteNode(root, x);

        printf("After deletion: ");
        inorderTraversal(temp);
        printf("\n");

        return 0;
    }
```

Status : Correct   ***Marks : 10/10***