Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

Answer

1, 4, 2, 18, 14, 13

Status : Correct ***Marks : 1/1***

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

Answer

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

Status : Correct ***Marks : 1/1***

3. 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***

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

Answer

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

Status : Correct ***Marks : 1/1***

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

Answer

13, 2, 1, 4, 14, 18

Status : Correct ***Marks : 1/1***

6. 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***

7. 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***

8. 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***

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

Answer

14

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. 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. 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, 4, 8, 5, 3, 11, 9

Status : Wrong ***Marks : 0/1***

14. 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***

15. 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***

Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
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 == 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* node) {
    while (node->left != NULL)
        node = node->left;
    return node;
}

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: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

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

```
// You are using GCC
#include <iostream>
using namespace std;

struct Node {
    int key;
    Node* left;
    Node* right;
    Node(int val) : key(val), left(NULL), right(NULL) {}
};

Node* insert(Node* root, int key) {
    if (!root) return new Node(key);
    if (key < root->key) root->left = insert(root->left, key);
    else root->right = insert(root->right, key);
    return root;
}

bool search(Node* root, int target) {
    if (!root) return false;
    if (root->key == target) return true;
    if (target < root->key) return search(root->left, target);
    else return search(root->right, target);
}
```

```
int main() {
    Node* root = NULL;
    int num;

    while (cin >> num && num != -1) {
        root = insert(root, num);
    }

    int target;
    cin >> target;

    if (search(root, target)) {
        cout << target << " is found in the BST" << endl;
    } else {
        cout << target << " is not found in the BST" << endl;
    }

    return 0;
}
```

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

2. 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 <iostream>
#include <queue>
using namespace std;

struct Node {
    int key;
    Node* left;
    Node* right;
    Node(int val) : key(val), left(NULL), right(NULL) {}
```

```
};

Node* insert(Node* root, int key) {
    if (!root) return new Node(key);
    if (key < root->key) root->left = insert(root->left, key);
    else root->right = insert(root->right, key);
    return root;
}

Node* minValueNode(Node* node) {
    while (node && node->left) node = node->left;
    return node;
}

Node* deleteNode(Node* root, int key) {
    if (!root) return NULL;
    if (key < root->key) root->left = deleteNode(root->left, key);
    else if (key > root->key) root->right = deleteNode(root->right, key);
    else {
        if (!root->left) {
            Node* temp = root->right;
            delete root;
            return temp;
        }
        if (!root->right) {
            Node* temp = root->left;
            delete root;
            return temp;
        }
        Node* temp = minValueNode(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
    return root;
}

void levelOrder(Node* root) {
    if (!root) return;
    queue<Node*> q;
    q.push(root);
    while (!q.empty()) {
        Node* curr = q.front(); q.pop();
```

```
        cout << curr->key << " ";
        if (curr->left) q.push(curr->left);
        if (curr->right) q.push(curr->right);
    }
    cout << endl;
}

int main() {
    int N, X, Y;
    cin >> N;
    Node* root = NULL;
    for (int i = 0; i < N; ++i) {
        int val;
        cin >> val;
        root = insert(root, val);
    }
    cin >> X >> Y;

    cout << "Initial BST: ";
    levelOrder(root);

    root = insert(root, X);
    cout << "BST after inserting a new node " << X << ": ";
    levelOrder(root);

    root = deleteNode(root, Y);
    cout << "BST after deleting node " << Y << ": ";
    levelOrder(root);

    return 0;
}
```

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

3. 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 key) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->key = key;
```

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

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

struct Node* trimBST(struct Node* root, int min, int max) {
    if (root == NULL) return NULL;
    root->left = trimBST(root->left, min, max);
    root->right = trimBST(root->right, min, max);
    if (root->key < min) {
        struct Node* rightChild = root->right;
        free(root);
        return rightChild;
    }
    if (root->key > max) {
        struct Node* leftChild = root->left;
        free(root);
        return leftChild;
    }
    return root;
}

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

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

```
        int val;
        scanf("%d", &val);
        root = insert(root, val);
    }
    int min, max;
    scanf("%d %d", &min, &max);
    root = trimBST(root, min, max);
    inorder(root);
    printf("\n");
    return 0;
}
```

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

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>
#include <limits.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 = 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 {
        root->right = insert(root->right, data);
    }
    return root;
}

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

```
}

struct Node* constructBSTFromPreorder(int preorder[], int* index, int n, int min,
int max) {
    if (*index >= n) {
        return NULL;
    }

    int data = preorder[*index];
    if (data < min || data > max) {
        return NULL;
    }

    (*index)++;
    struct Node* node = newNode(data);
    node->left = constructBSTFromPreorder(preorder, index, n, min, data);
    node->right = constructBSTFromPreorder(preorder, index, n, data, max);

    return node;
}

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

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

    int index = 0;
    struct Node* root = constructBSTFromPreorder(preorder, &index, n, INT_MIN,
INT_MAX);

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

    return 0;
}
```

Status : Correct *Marks : 10/10*

5. 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>

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

    return root;
}

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

    postOrder(root->left);
    postOrder(root->right);
    printf("%d ", root->data);
}

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

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

    struct Node* root = NULL;
```

```
    for (int i = 0; i < n; i++) {
        root = insert(root, elements[i]);
    }

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

    return 0;
}
```

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