Rajalakshmi Engineering College

Name: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

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

Write a program to help him perform the delete operation.

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
5 3 8 2 4 6
Output: 3 4 5 6 8

Answer

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


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


struct Node* 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
        root->right = insert(root->right, data);
    return root;
}

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

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


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

int main() {
    int N, i, val;
    scanf("%d", &N);
    struct Node* root = NULL;

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

```
    root = deleteMin(root);
    inorder(root);
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>

// Define a structure for BST nodes
typedef struct Node {
    int value;
    struct Node* left;
    struct Node* right;
} Node;

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

// Function to insert a value into the BST
Node* insert(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;
}

// Function to add a given value to all nodes in the BST
void addToAll(Node* root, int addValue) {
    if (root != NULL) {
        root->value += addValue;
        addToAll(root->left, addValue);
        addToAll(root->right, addValue);
    }
```

```
}

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

int main() {
    int N, addValue;

    // Read number of elements
    scanf("%d", &N);

    Node* root = NULL;

    // Read BST elements
    for (int i = 0; i < N; i++) {
        int element;
        scanf("%d", &element);
        root = insert(root, element);
    }

    // Read value to be added
    scanf("%d", &addValue);

    // Add value to all nodes
    addToAll(root, addValue);

    // Print maximum value in the BST after addition
    printf("%d\n", findMax(root));

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

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

```c
#include <stdlib.h>

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

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

Node* insert(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;
}

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

Node* deleteNode(Node* root, int key) {
    if (root == NULL) {
        return NULL;
    }

    if (key < root->value) {
        root->left = deleteNode(root->left, key);
```

```
    } else if (key > root->value) {
        root->right = deleteNode(root->right, key);
    } else {
        // Node with only one child or no child
        if (root->left == NULL) {
            Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            Node* temp = root->left;
            free(root);
            return temp;
        }

        // Node with two children, replace with inorder successor
        Node* temp = findMin(root->right);
        root->value = temp->value;
        root->right = deleteNode(root->right, temp->value);
    }
    return root;
}

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

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

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

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

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

    root = deleteNode(root, key);

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

return 0;
}
```

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

Rajalakshmi Engineering College

Name: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

int findMax(struct TreeNode* root) {
```

```
    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: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
    }
    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) {
    while (root->left != NULL) {
        root = root->left;
    }
    return 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);
    }

    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: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
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 {
        root->right = insertNode(root->right, value);
    }
    return root;
}

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

```
    else{
        return searchNode(root->right,searchValue);
    }
    }
```

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

Rajalakshmi Engineering College

Name: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
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: jayanthi v
Email: 240801131@rajalakshmi.edu.in
Roll no: 240801131
Phone: 9487003327
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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 value) {
    if (root == NULL) {
        return createNode(value);
```

```
    }
    if (value < root->data) {
        root->left = insert(root->left, value);
    } else {
        root->right = insert(root->right, value);
    }
    return root;
}
struct TreeNode* findMin(struct TreeNode* root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}
struct TreeNode* deleteNode(struct TreeNode* root, int value) {
    if (root == NULL) {
        return NULL;
    }
    if (value < root->data) {
        root->left = deleteNode(root->left, value);
    } else if (value > root->data) {
        root->right = deleteNode(root->right, value);
    } 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) {
        return;
```

```
    }
    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***