Rajalakshmi Engineering College

Name: Jeffery Antony J
Email: 241901041@rajalakshmi.edu.in
Roll no: 241901041
Phone: 7305663808
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

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

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


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


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

Node* insert(Node* root, char 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;
}


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


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


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

    Node* root = NULL;
    char value;

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

    char minVal = findMin(root);
    char maxVal = findMax(root);

    printf("Minimum value: %c\n", minVal);
    printf("Maximum value: %c\n", maxVal);

    return 0;
}
```

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

2. 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

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

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


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

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


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->data)
        root->left = deleteNode(root->left, key);
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    else {

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

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

Node* deleteMin(Node* root) {
    if (root == NULL) return NULL;

    Node* minNode = findMin(root);
    return deleteNode(root, minNode->data);
}

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

    Node* root = NULL;
    int val;

    for (int i = 0; i < N; i++) {
```

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

    root = deleteMin(root);

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

    return 0;
}
```

Status : Correct *Marks : 10/10*

3. 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

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


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


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


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


void addValue(Node* root, int add) {
```

```
    if (root == NULL) return;
    root->data += add;
    addValue(root->left, add);
    addValue(root->right, add);
}


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

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

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

    scanf("%d", &add);


    addValue(root, add);


    printf("%d\n", findMax(root));

    return 0;
}
```

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