Rajalakshmi Engineering College

Name: Phaveen S
Email: 240701383@rajalakshmi.edu.in
Roll no: 240701383
Phone: null
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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 key;
    struct Node* left;
    struct Node* right;
} Node;

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

Node* insertNode(Node* root, char key) {
    if (!root) return createNode(key);
    if (key < root->key)
        root->left = insertNode(root->left, key);
    else if (key > root->key)
        root->right = insertNode(root->right, key);
    return root;
}
```

```
char findMin(Node* root) {
    if (!root) return '\0';
    while (root->left)
        root = root->left;
    return root->key;
}

char findMax(Node* root) {
    if (!root) return '\0';
    while (root->right)
        root = root->right;
    return root->key;
}

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

    Node* root = NULL;
    for (int i = 0; i < N; i++) {
        char ch;
        scanf(" %c", &ch);
        root = insertNode(root, ch);
    }

    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

You are given a series of magic levels (integers) and need to construct a Binary Search Tree (BST) from them. After constructing the BST, your task

is to perform a range search, which involves finding and printing all the magic levels within a specified range [L, R].

Input Format

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

The second line consists of N space-separated integers, representing the magic levels to insert.

The third line consists of two integers, L and R, which define the range for the search.

Output Format

The output prints all the magic levels within the range [L, R] in ascending order, separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 3 7
2 20
Output: 3 5 7 10 15

Answer

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

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

Node* createNode(int key) {
    Node* newNode = (Node*)malloc(sizeof(Node));
```

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

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

void rangeSearch(Node* root, int L, int R) {
    if (!root) return;

    if (root->key > L)
        rangeSearch(root->left, L, R);

    if (root->key >= L && root->key <= R)
        printf("%d ", root->key);

    if (root->key < R)
        rangeSearch(root->right, L, R);
}

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

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

    int L, R;
    scanf("%d %d", &L, &R);

    rangeSearch(root, L, R);
```

```
    printf("\n");

    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

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

typedef struct Node {
    int key;
    struct Node *left, *right;
} Node;

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

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

void inOrderTraversal(Node* root) {
    if (!root) return;
    inOrderTraversal(root->left);
    printf("%d ", root->key);
    inOrderTraversal(root->right);
}

Node* findMin(Node* node) {
```

```
    while (node && node->left)
        node = node->left;
    return node;
}

Node* deleteNode(Node* root, int key, int* found) {
    if (!root) return NULL;

    if (key < root->key)
        root->left = deleteNode(root->left, key, found);
    else if (key > root->key)
        root->right = deleteNode(root->right, key, found);
    else {
        *found = 1;
        if (!root->left) {
            Node* temp = root->right;
            free(root);
            return temp;
        }
        else if (!root->right) {
            Node* temp = root->left;
            free(root);
            return temp;
        }
        Node* temp = findMin(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key, found);
    }
    return root;
}

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

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

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

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

    int found = 0;
    root = deleteNode(root, x, &found);

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

    return 0;
}
```

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