Rajalakshmi Engineering College

Name: SHREYA KS
Email: 240701506@rajalakshmi.edu.in
Roll no: 240701506
Phone: 9789293683
Branch: REC
Department: I CSE FE
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

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

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

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

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

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

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

int main() {
    int N;
    scanf("%d", &N);
    struct Node* root = NULL;
    char ch;
    for (int i = 0; i < N; i++) {
        scanf(" %c", &ch);  // Note space before %c to consume whitespace
        root = insert(root, ch);
    }

    printf("Minimum value: %c\n", findMin(root));
    printf("Maximum value: %c\n", findMax(root));
    return 0;
}
```

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

2. Problem Statement

John is building a system to store and manage integers using a binary search tree (BST). He needs to add a feature that allows users to search for a specific integer key in the BST using recursion.

Implement functions to create the BST and perform a recursive search for an integer.

Input Format

The first line of input consists of an integer representing, the number of nodes.

The second line consists of integers representing, the values of nodes, separated by space.

The third line consists of an integer representing, the key to be searched.

Output Format

The output prints whether the given key is present in the binary search tree or not.

Refer to the sample output for the exact format.

Sample Test Case

Input: 7
10 5 15 3 7 12 20
12
Output: The key 12 is found in the binary search tree

Answer

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

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

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

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

int search(struct Node* root, int key) {
    if (root == NULL) return 0;
    if (root->data == key) return 1;
```

```
    if (key < root->data) return search(root->left, key);
    return search(root->right, key);
}

int main() {
    int n, val, key;
    scanf("%d", &n);
    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d", &key);
    if (search(root, key))
        printf("The key %d is found in the binary search tree\n", key);
    else
        printf("The key %d is not found in the binary search tree\n", key);
    return 0;
}
```

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

3. 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

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

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

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

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

void rangeSearch(struct Node* root, int L, int R) {
    if (root == NULL) return;
    if (root->data >= L && root->data <= R) {
        rangeSearch(root->left, L, R);
```

```
        printf("%d ", root->data);
        rangeSearch(root->right, L, R);
    } else if (root->data < L) {
        rangeSearch(root->right, L, R);
    } else {
        rangeSearch(root->left, L, R);
    }
}

int main() {
    int N, val, L, R;
    scanf("%d", &N);
    struct Node* root = NULL;
    for (int i = 0; i < N; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d %d", &L, &R);
    rangeSearch(root, L, R);
    printf("\n");
    return 0;
}
```

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