Rajalakshmi Engineering College

Name: shobbika T
Email: 240701502@rajalakshmi.edu.in
Roll no: 240701502
Phone: 7305423247
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

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.

240701502

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>

// Define structure for BST node
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

// Create a new BST node
struct Node* newNode(int data) {
   struct Node* node = (struct Node*)malloc(sizeof(struct Node));
   node->data = data;
   node->left = node->right = NULL;
   return node;
}

// Insert into BST
struct Node* insert(struct Node* root, int data) {
   if (root == NULL)
      return newNode(data);
   if (data < root->data)
      root->left = insert(root->left, data);
```

240701502

```
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}

// In-order traversal with range filter
void rangeSearch(struct Node* root, int L, int R) {
    if (root == NULL) return;
    if (root->data > L)
        rangeSearch(root->left, L, R);
    if (root->data >= L && root->data <= R)
        printf("%d ", root->data);
    if (root->data < R)
        rangeSearch(root->right, L, R);
}

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

    // Read and insert N magic levels
    for (int i = 0; i < N; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }

    // Read range
    scanf("%d %d", &L, &R);

    // Perform range search and print result
    rangeSearch(root, L, R);
    printf("\n");

    return 0;
}
```

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

2. Problem Statement

240701502

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;
   struct Node* right;
};
struct Node* newNode(char data) {
   struct Node* node = (struct Node*)malloc(sizeof(struct Node));
```

240701502

```
    node->data = data;
    node->left = node->right = NULL;
    return node;
}
struct Node* insert(struct Node* root, char 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;
}
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);
    char ch;
    struct Node* root = NULL;
    for (int i = 0; i < N; i++) {
        scanf(" %c", &ch);
        root = insert(root, ch);
    }
    char minVal = findMin(root);
    char maxVal = findMax(root);

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

    return 0;
}
```

240701502

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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
```

240701502

```
    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 if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}
void addToAllNodes(struct Node* root, int add) {
    if (root == NULL) return;
    root->data += add;
    addToAllNodes(root->left, add);
    addToAllNodes(root->right, add);
}
int findMax(struct Node* root) {
    while (root->right != NULL)
        root = root->right;
    return root->data;
}
int main() {
    int N, val, add;
    scanf("%d", &N);
    struct Node* root = NULL;
    for (int i = 0; i < N; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d", &add);
    addToAllNodes(root, add);
    int maxVal = findMax(root);
    printf("%d\n", maxVal);
```

240701502

```
    return 0;
}
```

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

240701502