Rajalakshmi Engineering College

Name: Naren S
Email: 241901066@rajalakshmi.edu.in
Roll no: 241901066
Phone: 6382463115
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

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

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

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

struct Node* insert(struct Node* root, char 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;
}
```

```
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;
   for (int i = 0; i < n; i++) {
      char ch;
      scanf(" %c", &ch);
      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

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 val;
    struct Node* left;
    struct Node* right;
} Node;

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

Node* insert(Node* root, int val) {
    if (!root) return newNode(val);
    if (val < root->val)
        root->left = insert(root->left, val);
    else if (val > root->val)
```

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

void addToAllNodes(Node* root, int add) {
    if (!root) return;
    root->val += add;
    addToAllNodes(root->left, add);
    addToAllNodes(root->right, add);
}

int findMax(Node* root) {
    while (root && root->right)
        root = root->right;
    return root ? root->val : -1;
}

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);
    addToAllNodes(root, add);
    printf("%d", findMax(root));
    return 0;
}
```

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

3. Problem Statement

Edward has a Binary Search Tree (BST) and needs to find the k-th largest element in it.

Given the root of the BST and an integer k, help Edward determine the k-th largest element in the tree. If k exceeds the number of nodes in the BST, return an appropriate message.

Input Format

The first line of input consists of integer n, the number of nodes in the BST.

The second line consists of the n elements, separated by space.

The third line consists of the value of k.

Output Format

The output prints the kth largest element in the binary search tree.

For invalid inputs, print "Invalid value of k".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 4 12 2 6 10 14
1
Output: 14

Answer

```
// You are using GCC
#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 = NULL;
   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;
}

void reverseInorder(struct Node* root, int k, int* count, int* result) {
    if (root == NULL || *count >= k) return;
    reverseInorder(root->right, k, count, result);
    (*count)++;
    if (*count == k) {
        *result = root->data;
        return;
    }
    reverseInorder(root->left, k, count, result);
}

int countNodes(struct Node* root) {
    if (root == NULL) return 0;
    return 1 + countNodes(root->left) + countNodes(root->right);
}

int main() {
    int n, k, val;
    scanf("%d", &n);
    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d", &k);
    int total = countNodes(root);
    if (k > total || k <= 0) {
        printf("Invalid value of k\n");
    } else {
        int count = 0, result = -1;
        reverseInorder(root, k, &count, &result);
        printf("%d\n", result);
    }
    return 0;
}
```

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