Rajalakshmi Engineering College

Name: Rithesh Madhav S
Email: 240701428@rajalakshmi.edu.in
Roll no: 240701428
Phone: 9884267696
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>


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 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() {
    struct Node* root = NULL;
    int N;
    char value;


    scanf("%d", &N);


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


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

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

Node* insert(Node* root, int data) {
    if (root == NULL) return createNode(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* deleteMin(Node* root) {
    if (root == NULL) return NULL;
    if (root->left == NULL) {
        Node* rightSubtree = root->right;
        free(root);
        return rightSubtree;
    }
    root->left = deleteMin(root->left);
    return root;
}

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

int main() {
    int N, i, data;
    scanf("%d", &N);
    Node* root = NULL;
    for (i = 0; i < N; i++) {
```

```
        scanf("%d", &data);
        root = insert(root, data);
    }
    root = deleteMin(root);
    inorder(root);
    return 0;
}
```

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

3. 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

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

int main() {
```

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