Rajalakshmi Engineering College

Name: KEERTHI PRIYA T
Email: 240701258@rajalakshmi.edu.in
Roll no: 2116240701258
Phone: 7397397221
Branch: REC
Department: I CSE FC
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

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.

2116240701258

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>

// Structure for each node in the BST
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

// Function to create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a value into the BST
struct Node* insert(struct Node* root, int value) {
    if (root == NULL) return createNode(value);
    if (value < root->data)
        root->left = insert(root->left, value);
```

2116240701258

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

// Function to perform in-order traversal
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

// Function to delete the minimum element from BST
struct Node* deleteMin(struct Node* root) {
    if (root == NULL) return NULL;
    if (root->left == NULL) {
        struct Node* temp = root->right;
        free(root);
        return temp;
    }
    root->left = deleteMin(root->left);
    return root;
}

// Main function
int main() {
    int n, value;
    scanf("%d", &n);
    struct Node* root = NULL;

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

    root = deleteMin(root);

    inorder(root);
    printf("\n");
```

2116240701258

```
    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>
```

2116240701258

```
#include <stdlib.h>

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

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

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

void rangeSearch(struct Node* root, int L, int R) {
   if (root == NULL) return;

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

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

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

int main() {
   int n, value, L, R;
   scanf("%d", &n);

   struct Node* root = NULL;
```

2116240701258

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

    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

2116240701258

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>

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


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


struct Node* insert(struct Node* root, int value) {
   if (root == NULL) return createNode(value);
   if (value < root->data)
      root->left = insert(root->left, value);
   else if (value > root->data)
      root->right = insert(root->right, value);
```

2116240701258

```
    return root;
}


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


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


struct Node* deleteNode(struct Node* root, int key) {
    if (root == NULL) return NULL;

    if (key < root->data) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->data) {
        root->right = deleteNode(root->right, key);
    } else {

        if (root->left == NULL) {
            struct Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct Node* temp = root->left;
            free(root);
            return temp;
        }

        struct Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
```

2116240701258

```
    return root;
}

int search(struct Node* root, int key) {
    if (root == NULL) return 0;
    if (key == root->data) return 1;
    if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}

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

    struct Node* root = NULL;

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

    scanf("%d", &x);

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

    if (search(root, x)) {
        root = deleteNode(root, x);
    }


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

    return 0;
}
```

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

2116240701258