Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

240701215

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.

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 key;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int key) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->key = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}
Node* insert(Node* root, int key) {
    if (root == NULL) return createNode(key);
    if (key < root->key)
        root->left = insert(root->left, key);
    else
        root->right = insert(root->right, key);
    return root;
}
Node* findMin(Node* root) {
```

240701215

240701215

```
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}
Node* deleteMin(Node* root) {
    if (root == NULL) return NULL;

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

    root->left = deleteMin(root->left);
    return root;
}
void inOrder(Node* root) {
    if (root == NULL) return;
    inOrder(root->left);
    printf("%d ", root->key);
    inOrder(root->right);
}

int main() {
    int N;
    scanf("%d", &N);
    int data[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &data[i]);
    }

    Node* root = NULL;
    for (int i = 0; i < N; i++) {
        root = insert(root, data[i]);
    }
    root = deleteMin(root);
    inOrder(root);
    printf("\n");
    return 0;
}
```

240701215

Status : Correct *Marks : 10/10*

2. 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>

typedef struct Node {
   char key;
```

```
    struct Node* left;
    struct Node* right;
} Node;

// Function to create a new tree node
Node* createNode(char key) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->key = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a node in BST
Node* insert(Node* root, char key) {
    if (root == NULL) return createNode(key);
    if (key < root->key)
        root->left = insert(root->left, key);
    else
        root->right = insert(root->right, key);
    return root;
}

// Function to find the minimum value node in BST
Node* findMin(Node* root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}

// Function to find the maximum value node in BST
Node* findMax(Node* root) {
    while (root->right != NULL) {
        root = root->right;
    }
    return root;
}

int main() {
    int N;

    // Input for the number of elements to insert into the BST
```

240701215

240701215

```
    scanf("%d", &N);
    char data[N];

    // Input for the characters to be inserted into the BST
    for (int i = 0; i < N; i++) {
        scanf(" %c", &data[i]); // Notice the space before %c to consume any
whitespace
    }

    Node* root = NULL;

    // Insert elements into the BST
    for (int i = 0; i < N; i++) {
        root = insert(root, data[i]);
    }

    // Find and print the minimum and maximum values
    Node* minNode = findMin(root);
    Node* maxNode = findMax(root);

    printf("Minimum value: %c\n", minNode->key);
    printf("Maximum value: %c\n", maxNode->key);

    // Free allocated memory (not implemented for simplicity)
    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.

240701215

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

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

typedef struct Node {
   int key;
   struct Node* left;
   struct Node* right;
} Node;
Node* createNode(int key) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->key = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
Node* insert(Node* root, int key) {
   if (root == NULL) return createNode(key);
   if (key < root->key)
      root->left = insert(root->left, key);
   else
      root->right = insert(root->right, key);
```

```
    return root;
}
int count = 0;
int kthLargestValue = -1;
void reverseInOrder(Node* root, int k) {
    if (root == NULL || count >= k) return;
    reverseInOrder(root->right, k);
    count++;
    if (count == k) {
        kthLargestValue = root->key;
        return;
    }
    reverseInOrder(root->left, k);
}

int main() {
    int n, k;
    scanf("%d", &n);
    int data[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &data[i]);
    }
    scanf("%d", &k);
    Node* root = NULL;
    for (int i = 0; i < n; i++) {
        root = insert(root, data[i]);
    }
    if (k < 1 || k > n) {
        printf("Invalid value of k\n");
    } else {
        reverseInOrder(root, k);
        printf("%d\n", kthLargestValue);
    }
    return 0;
}
```

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

240701215