Rajalakshmi Engineering College

Scan to verify results

Name: Anisha .
Email: 241801023@rajalakshmi.edu.in
Roll no:241801023
Phone: 9003003291
Branch: REC
Department: l AI & DS
FA Batch: 2028
Degree: B.E - AI & DS

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is learning about Binary Search Trees (BST) in his computer science class. He wants to create a program that allows users to delete a node with a given value from a BST and print the remaining nodes using an in-order traversal.

Implement a function to help him delete a node with a given value from a BST.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the BST nodes.

The third line consists of an integer V, which is the value to delete from the BST.

Output Format

The output prints the space-separated values in the BST in an in-order traversal, after the deletion of the specified value.

If the specified value is not available in the tree, print the given input values in-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7
15

Output: 2 5 7 10

Answer

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

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

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


// Insert a node into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
   if (root == NULL) {
      struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
      newNode->data = key;
      newNode->left = newNode->right =
      NULL; return newNode;
   }
   if (key < root->data)
      root->left = insert(root->left,
   key); else if (key > root->data)
      root->right = insert(root->right,
   key); return root;
}

// Find the minimum value node in a subtree
struct TreeNode* findMin(struct TreeNode* root)
{
   while (root->left != NULL)
      root = root->left;
   return root;
}

// Delete a node with a given key from the BST
struct TreeNode* deleteNode(struct TreeNode* root, int key)
   { if (root == NULL)
      return root;

   if (key < root->data)
      root->left = deleteNode(root->left, key);
   else if (key > root->data)
      root->right = deleteNode(root->right,
   key); else {
      // Node with one or no
      child if (root->left == NULL)
      {
         struct TreeNode* temp = root->right;
         free(root);
         return temp;
      } else if (root->right == NULL) {
         struct TreeNode* temp = root->left;
         free(root);
         return temp;
      }
```

```
        // Node with two children
        struct TreeNode* temp =
        findMin(root->right); root->data =
        temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

// In-order traversal of BST
void inorderTraversal(struct TreeNode* root)
    { if (root == NULL)
        return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorde
    rTraversal(r
    oot->right);
}



int main()
{
    int N, rootValue, V;
    scanf("%d", &N);
    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }
    scanf("%d", &V);
    root = deleteNode(root,
    V); inorderTraversal(root);
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Anisha .
Email: 241801023@rajalakshmi.edu.in
Roll no:241801023
Phone: 9003003291
Branch: REC
Department: I AI & DS
FA Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

Input Format

The first line of input consists of an integer N, representing the number of values to insert into the BST.

The second line consists of N space-separated integers, representing the values to insert into the BST.

Output Format

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 1 5 2 4

Output: 3 1 2 5 4

Answer

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

// Preorder traversal: Root -> Left -> Right
```

```
void printPreorder(struct Node* node)
   { if (node == NULL)
      return;
   printf("%d ", node->data);
   printPreorder(node->left);
   printP
   reorder(nod
   e->right);
}


int main() {
   struct Node* root = NULL;

   int n;
   scanf("%d",
   &n);

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

   printPreorder(root);
   return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Anisha .
Email: 241801023@rajalakshmi.edu.in
Roll no:241801023
Phone: 9003003291
Branch: REC
Department: I AI & DS
FA Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

Input Format

The first line of input consists of an integer n, representing the number of nodes in the binary search tree.

The second line consists of the values of the nodes, separated by space as integers.

The third line consists of an integer representing, the value that is to be searched.

Output Format

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 3 10 1 6 14 23
6

Output: Value 6 is found in the 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 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
      root->right = insert(root->right, value);
   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,
   key;
   scanf("%d", &n);

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

   scanf("%d", &key);

   if (search(root, key))
      printf("Value %d is found in the tree.\n", key);
   else
      printf("Value %d is not found in the tree.\n", key);

   return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Anisha .
Email: 241801023@rajalakshmi.edu.in
Roll no:241801023
Phone: 9003003291
Branch: REC
Department: l AI & DS
FA Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15

Output: 15 10 5
The minimum value in the BST is: 5

Answer

```
#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;
}
void displayTreePostOrder(struct Node* root)
   { if (root == NULL) {
      return;
   }
   displayTreePostOrder(root->left);
   display
   TreePostOrde
   r(root->right); printf("%d ",
   root->data);
}
int findMinValue(struct Node* root)
   { if (root == NULL) {
      return -1;
   }
   while (root->left != NULL)
      { root = root->left;
   }
   return root->data;
}

int main() {
   struct Node* root = NULL;
   int n, data;
   scanf("%d", &n);

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

   displayTreePostOrder(root);
   printf("\n");

   int minValue = findMinValue(root);
   printf("The minimum value in the BST is: %d", minValue);

   return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Anisha .
Email: 241801023@rajalakshmi.edu.in
Roll no:241801023
Phone: 9003003291
Branch: REC
Department: I AI & DS
FA Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the nodes to insert into the BST.

Output Format

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7

Output: 15

Answer

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

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


struct TreeNode* insert(struct TreeNode* root, int key)
   { if (root == NULL) {
      return createNode(key);
   }

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

   return root;
}

// Function to find the maximum value in
BST int findMax(struct TreeNode* root) {
  if (root == NULL)
    { return -1;
  }

  struct TreeNode* current = root;
  while (current->right != NULL) {
    current = current->right;
  }
  return current->data;
}

int main() {
  int N,
  rootValue;
  scanf("%d",
  &N);

  struct TreeNode* root = NULL;

  for (int i = 0; i < N; i++) {
    int key;
    scanf("%d", &key);
    if (i == 0) rootValue = key;
    root = insert(root, key);
  }

  int maxVal =
  findMax(root); if
  (maxVal != -1) {
    printf("%d", maxVal);
  }

  return 0;
}
```

Status : Correct *Marks : 10/10*