Rajalakshmi Engineering College

Name: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. Which of the following operations can be used to traverse a Binary Search Tree (BST) in ascending order?

Answer

Inorder traversal

Status : Correct ***Marks : 1/1***

2. Which of the following is the correct post-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

20, 32, 30, 52, 57, 55, 50

Status : Correct ***Marks : 1/1***

3. While inserting the elements 5, 4, 2, 8, 7, 10, 12 in a binary search tree, the element at the lowest level is _________.

Answer

12

Status : Correct ***Marks : 1/1***

4. Find the pre-order traversal of the given binary search tree.

Answer

13, 2, 1, 4, 14, 18

Status : Correct ***Marks : 1/1***

5. Which of the following is the correct in-order traversal of a binary search tree with nodes: 9, 3, 5, 11, 8, 4, 2?

Answer

2, 3, 4, 5, 8, 9, 11

Status : Correct ***Marks : 1/1***

6. In a binary search tree with nodes 18, 28, 12, 11, 16, 14, 17, what is the value of the left child of the node 16?

Answer

14

Status : Correct ***Marks : 1/1***

7. The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19. Which one of the following is the postorder traversal of the tree?

Answer

11, 12, 10, 16, 19, 18, 20, 15

Status : Correct ***Marks : 1/1***

8. How many distinct binary search trees can be created out of 4 distinct keys?

Answer

14

Status : Correct ***Marks : 1/1***

9. Which of the following is the correct pre-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

50, 30, 20, 32, 55, 52, 57

Status : Correct ***Marks : 1/1***

10. Find the in-order traversal of the given binary search tree.

Answer

1, 2, 4, 13, 14, 18

Status : Correct ***Marks : 1/1***

11. While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is _______.

Answer

67

Status : Correct ***Marks : 1/1***

12. Which of the following is a valid preorder traversal of the binary search tree with nodes: 18, 28, 12, 11, 16, 14, 17?

Answer

18, 12, 11, 16, 14, 17, 28

Status : Correct ***Marks : 1/1***

13. Find the postorder traversal of the given binary search tree.

Answer

1, 4, 2, 18, 14, 13

Status : Correct ***Marks : 1/1***

14. Find the preorder traversal of the given binary search tree.

Answer

9, 2, 1, 6, 4, 7, 10, 14

Status : Correct ***Marks : 1/1***

15. Find the post-order traversal of the given binary search tree.

Answer

10, 17, 20, 18, 15, 32, 21

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 2
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;
}
// You are using GCC
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=NULL;
        newNode->right=NULL;
        return newNode;
    }else if(key<root->data){
        root->left=insert(root->left,key);
    }else if(key>root->data){
        root->right=insert(root->right,key);
    }
    return root;
}

struct TreeNode* findMin(struct TreeNode* root) {
    while(root&&root->left){
        root=root->left;
    }
    return root;
}

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{
        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;
        }
        struct TreeNode*temp = findMin(root->right);
        root->data=temp->data;
        root->right=deleteNode(root->right,temp->data);
    }
    return root;
}

void inorderTraversal(struct TreeNode* root) {
    if (root!=NULL){
        inorderTraversal(root->left);
        printf("%d\t",root->data);
        inorderTraversal(root->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: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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;
}
// You are using GCC
struct Node* insert(struct Node* root, int value) {
   if (root==NULL){
      struct Node*newNode = (struct Node*)malloc(sizeof(struct Node));
      newNode->data = value;
      newNode->left = NULL;
      newNode->right = NULL;
      return newNode;
   }
   if(value<root->data){
      root->left = insert(root->left,value);
   }else{
        root->right = insert(root->right,value);
    }
    return root;
    }
    void printPreorder(struct Node* node) {
      if (node == NULL)
      return;
      printf("%d", node->data);
      printPreorder(node->left);
      printPreorder(node->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: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

```
#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));
   if(!newNode){
      printf("Memory allocation error!\n");
      return NULL;
   }
   newNode->data=value;
   newNode->left=newNode->right=NULL;
   return newNode;
}
struct Node*insertNode(struct Node*root,int value){
   if(root==NULL){
        return createNode(value);
    }
    if (value<root->data){
        root->left=insertNode(root->left,value);
    }else if(value>root->data){
        root->right=insertNode(root->right,value);
    }
    return root;
}
struct Node*searchNode(struct Node* root,int value){
    if(root==NULL||root->data==value){
        return root;
    }
    if (value<root->data){
        return searchNode(root->left,value);
    }else{
        return searchNode(root->right,value);
    }
}
int main(){
    struct Node*root=NULL;
    int numNodes,value,searchValue;
    scanf("%d",&numNodes);
    for(int i=0;i<numNodes;i++){
        scanf("%d",&value);
        root=insertNode(root,value);
    }
    scanf("%d",&searchValue);
    struct Node*searchResult=searchNode(root,searchValue);
    if(searchResult!=NULL){
        printf("Value %d is found in the tree.\n",searchValue);
    }else{
        printf("Value %d is not found in the tree.\n",searchValue);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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;
}

// You are using GCC
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 displayTreePostOrder(struct Node* root) {
    if(root==NULL){
        return;
    }
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d",root->data);
}

int findMinValue(struct Node* root) {
    struct Node*current =root;
    while(current && current->left!=NULL){
        current=current->left;
    }
    return current->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: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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;
}

// You are using GCC
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;
}

int findMax(struct TreeNode* root) {
   if (root == NULL) {
      return -1;

   }
   while(root->right !=NULL){
      root=root->right;
   }
   return root->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***

Rajalakshmi Engineering College

Name: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
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

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 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

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

// Define the structure of a node
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

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

```
// Insert into BST
struct Node* insert(struct Node* root, int value) {
   if (root == NULL)
      return newNode(value);
   if (value < root->data)
      root->left = insert(root->left, value);
   else if (value > root->data)
      root->right = insert(root->right, value);
   return root;
}

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

// Find minimum value node
struct Node* findMin(struct Node* node) {
   while (node && node->left != NULL)
      node = node->left;
   return node;
}

// Delete node from BST
struct Node* deleteNode(struct Node* root, int key, int* found) {
   if (root == NULL)
      return root;

   if (key < root->data)
      root->left = deleteNode(root->left, key, found);
   else if (key > root->data)
      root->right = deleteNode(root->right, key, found);
   else {
      *found = 1;
      // Node with only one child or no child
      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;
        }

        // Node with two children: Get inorder successor
        struct Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data, found);
    }
    return root;
}

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

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

    scanf("%d", &x);

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

    int found = 0;
    root = deleteNode(root, x, &found);

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

    return 0;
```

```
}
```

Status : Correct *Marks : 10/10*

2. 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

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

```
// Define structure for a BST node
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

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

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

// Recursive search function
int search(struct Node* root, int key) {
   if (root == NULL)
      return 0;
   if (key == root->data)
      return 1;
   else 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 val;
        scanf("%d", &val);
        root = insert(root, val);
    }

    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***

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

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

// Define the BST node structure
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

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

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

```
    return root;
}

// Reverse in-order traversal to find k-th largest
void kthLargestUtil(struct Node* root, int k, int* count, int* result) {
    if (root == NULL || *count >= k)
        return;
    kthLargestUtil(root->right, k, count, result);
    (*count)++;
    if (*count == k) {
        *result = root->data;
        return;
    }
    kthLargestUtil(root->left, k, count, result);
}

// Count total nodes in BST
int countNodes(struct Node* root) {
    if (root == NULL)
        return 0;
    return 1 + countNodes(root->left) + countNodes(root->right);
}

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

    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        int val;
        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;
        kthLargestUtil(root, k, &count, &result);
        printf("%d\n", result);
    }

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Pranathee H
Email: 241901078@rajalakshmi.edu.in
Roll no: 241901078
Phone: 8925695249
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Yogi is working on a program to manage a binary search tree (BST) containing integer values. He wants to implement a function that removes nodes from the tree that fall outside a specified range defined by a minimum and maximum value.

Help Yogi by writing a function that achieves this.

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, representing the elements to be inserted into the BST.

The third line consists of two space-separated integers min and max, representing the minimum value and the maximum value of the range.

Output Format

The output prints the remaining elements of the BST in an in-order traversal, after removing nodes that fall outside the specified range.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 12
5 15
Output: 5 10 12 15

Answer

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

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

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

// Insert a node in BST
struct Node* insert(struct Node* root, int data) {
   if (root == NULL) return newNode(data);
   if (data < root->data) root->left = insert(root->left, data);
   else root->right = insert(root->right, data);
   return root;
```

```
}

// Function to trim BST
struct Node* trimBST(struct Node* root, int min, int max) {
   if (root == NULL) return NULL;

   // First fix the left and right subtrees
   root->left = trimBST(root->left, min, max);
   root->right = trimBST(root->right, min, max);

   // Now fix the root
   if (root->data < min) {
      struct Node* rightChild = root->right;
      free(root);
      return rightChild;
   }

   if (root->data > max) {
      struct Node* leftChild = root->left;
      free(root);
      return leftChild;
   }

   return root;
}

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

// Main function
int main() {
   int n;
   scanf("%d", &n);

   struct Node* root = NULL;
   for (int i = 0; i < n; i++) {
      int val;
```

```
        scanf("%d", &val);
        root = insert(root, val);
    }

    int min, max;
    scanf("%d %d", &min, &max);

    root = trimBST(root, min, max);

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

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

2. Problem Statement

Aishu is participating in a coding challenge where she needs to reconstruct a Binary Search Tree (BST) from given preorder traversal data and then print the in-order traversal of the reconstructed BST.

Since Aishu is just learning about tree data structures, she needs your help to write a program that does this efficiently.

Input Format

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

The second line of input contains n integers separated by spaces, which represent the preorder traversal of the BST.

Output Format

The output displays n space-separated integers, representing the in-order traversal of the reconstructed BST.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
10 5 1 7 40 50

Output: 1 5 7 10 40 50

Answer

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

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

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

// Recursive function to construct BST from preorder
struct Node* buildBST(int preorder[], int* index, int n, int min, int max) {
   if (*index >= n)
      return NULL;

   int val = preorder[*index];

   if (val < min || val > max)
      return NULL;

   struct Node* root = newNode(val);
   (*index)++;

   root->left = buildBST(preorder, index, n, min, val - 1);
   root->right = buildBST(preorder, index, n, val + 1, max);

   return root;
}

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

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

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

    int index = 0;
    struct Node* root = buildBST(preorder, &index, n, INT_MIN, INT_MAX);

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

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

3. Problem Statement

Joseph, a computer science student, is interested in understanding binary search trees (BST) and their node arrangements. He wants to create a program to explore BSTs by inserting elements into a tree and displaying the nodes using post-order traversal of the tree.

Write a program to help Joseph implement 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 output prints N space-separated integer values after the post-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 15 5 3

Output: 3 5 15 10

Answer

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

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

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

// Function to insert data into the BST
struct Node* insert(struct Node* root, int data) {
   if (root == NULL)
      return newNode(data);
```

```
    if (data < root->data)
        root->left = insert(root->left, data);
    else
        root->right = insert(root->right, data);

    return root;
}

// Function to perform post-order traversal
void postOrder(struct Node* root) {
    if (root == NULL)
        return;

    postOrder(root->left);
    postOrder(root->right);
    printf("%d ", root->data);
}

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

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

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

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

4. Problem Statement

Arun is exploring operations on binary search trees (BST). He wants to write a program with an unsorted distinct integer array that represents the BST keys and construct a height-balanced BST from it.

After constructing, he wants to perform the following operations that can alter the structure of the tree and traverse them using a level-order traversal:

InsertionDeletion

Your task is to assist Arun in completing the program without any errors.

Input Format

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

The second line consists of N space-separated integers, representing the initial keys.

The third line consists of an integer X, representing the new key to be inserted into the BST.

The fourth line consists of an integer Y, representing the key to be deleted from the BST.

Output Format

The first line of output prints "Initial BST: " followed by a space-separated list of keys in the initial BST after constructing it in level order traversal.

The second line prints "BST after inserting a new node X: " followed by a space-separated list of keys in the BST after inserting X n level order traversal.

The third line prints "BST after deleting node Y: " followed by a space-separated list of keys in the BST after deleting Y n level order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
25 14 56 28 12
34
12

Output: Initial BST: 25 14 56 12 28
BST after inserting a new node 34: 25 14 56 12 28 34
BST after deleting node 12: 25 14 56 28 34

Answer

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

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

// Queue Node
struct QueueNode {
   struct Node* treeNode;
   struct QueueNode* next;
};

// Queue
struct Queue {
   struct QueueNode* front;
   struct QueueNode* rear;
};

struct Queue* createQueue() {
   struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
   q->front = q->rear = NULL;
   return q;
}

void enqueue(struct Queue* q, struct Node* node) {
   struct QueueNode* temp = (struct QueueNode*)malloc(sizeof(struct
QueueNode));
   temp->treeNode = node;
   temp->next = NULL;
   if (q->rear == NULL) {
      q->front = q->rear = temp;
      return;
   }
```

```
    q->rear->next = temp;
    q->rear = temp;
}

struct Node* dequeue(struct Queue* q) {
    if (q->front == NULL) return NULL;
    struct QueueNode* temp = q->front;
    q->front = q->front->next;
    if (q->front == NULL) q->rear = NULL;
    struct Node* node = temp->treeNode;
    free(temp);
    return node;
}

int isQueueEmpty(struct Queue* q) {
    return q->front == NULL;
}

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

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

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

struct Node* deleteNode(struct Node* 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 {
        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);
    }
    return root;
}

void levelOrder(struct Node* root) {
    if (root == NULL) return;
    struct Queue* q = createQueue();
    enqueue(q, root);
    while (!isQueueEmpty(q)) {
        struct Node* current = dequeue(q);
        printf("%d ", current->data);
        if (current->left) enqueue(q, current->left);
        if (current->right) enqueue(q, current->right);
    }
    free(q);
}

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

    int arr[100];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    int X, Y;
    scanf("%d %d", &X, &Y);

    // Build BST using insertion order (NOT sorted)
    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        root = insert(root, arr[i]);
    }

    printf("Initial BST: ");
    levelOrder(root);
    printf("\n");

    root = insert(root, X);
    printf("BST after inserting a new node %d: ", X);
    levelOrder(root);
    printf("\n");

    root = deleteNode(root, Y);
    printf("BST after deleting node %d: ", Y);
    levelOrder(root);
    printf("\n");

    return 0;
}
```

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

5. Problem Statement

Viha, a software developer, is working on a project to automate searching for a target value in a Binary Search Tree (BST). She needs to create a program that takes an integer target value as input and determines if that value is present in the BST or not.

Write a program to assist Viha.

Input Format

The first line of input consists of integers separated by spaces, which represent the elements to be inserted into the BST. The input is terminated by entering -1.

The second line consists of an integer target, which represents the target value to be searched in the BST.

Output Format

If the target value is found in the BST, print "[target] is found in the BST".

Else, print "[target] is not found in the BST"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 3 7 1 4 6 8 -1
4
Output: 4 is found in the BST

Answer

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

// Define a BST node
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

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

// Insert function for BST
struct Node* insert(struct Node* root, int data) {
   if (root == NULL)
      return newNode(data);
```

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

// Search function for BST
int search(struct Node* root, int target) {
    if (root == NULL)
        return 0;
    if (target == root->data)
        return 1;
    else if (target < root->data)
        return search(root->left, target);
    else
        return search(root->right, target);
}

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

    // Read elements until -1
    while (1) {
        scanf("%d", &value);
        if (value == -1)
            break;
        root = insert(root, value);
    }

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

    if (search(root, target))
        printf("%d is found in the BST\n", target);
    else
        printf("%d is not found in the BST\n", target);

    return 0;
}
```

Status : Correct *Marks : 10/10*