Rajalakshmi Engineering College

Name: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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 : 14

Section 1 : MCQ

1. 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***

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

Answer

14

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

3. 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*

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

Answer

1, 4, 2, 18, 14, 13

Status : Correct *Marks : 1/1*

5. 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

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

Status : Wrong *Marks : 0/1*

6. 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*

7. 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***

8. 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***

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

Answer

1, 2, 4, 13, 14, 18

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

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

Answer

13, 2, 1, 4, 14, 18

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

11. 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***

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

Answer

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

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

13. 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***

14. 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***

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

Answer

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

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

Rajalakshmi Engineering College

Name: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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 : 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;
}

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;
}

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

struct TreeNode* deleteNode(struct TreeNode* 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 && root->right){
            struct TreeNode* temp = findMin(root->right);
            root->data = temp->data;
            root->right = deleteNode(root->right,temp->data);
        }
        else{
            struct TreeNode* temp = root;
            if (root->left == NULL){
                root = root->right;
            }
```

```
            else {
                root = root->left;
                }
                free(temp);
            }
        }
        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: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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) {
    struct Node* nn = createNode(value);
    if (root == NULL){
        nn->data = value;
        nn->left = NULL;
        nn->right = NULL;
        root = nn;
    }
    else if (value<root->data){
        root->left = insert(root->left,value);
    }
```

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

void printPreorder(struct Node* node) {
    if (node!=NULL){
        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: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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

```
// You are using GCC

struct Node* insertNode(struct Node* root, int value) {
    if (root == NULL){
        struct Node* nn = createNode(value);
        root = nn;
    }
    else 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){
        return NULL;
    }
```

```
    else if (value<root->data){
        return searchNode(root->left,value);
    }
    else if (value>root->data){
        return searchNode(root->right,value);
    }
    else{
        return root;
    }
}
```

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

Rajalakshmi Engineering College

Name: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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){
        struct Node* nn = createNode(data);
        root = nn;
    }
    else 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){
        displayTreePostOrder(root->left);
        displayTreePostOrder(root->right);
        printf("%d ", root->data);
    }
}

int findMinValue(struct Node* root) {
    struct Node* curr = root;
    while(curr->left!=NULL){
        curr=curr->left;
    }
    return curr->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: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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){
        struct TreeNode* nn = createNode(key);
        root = nn;
    }
    else 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) {
    struct TreeNode* curr = root;
    while (curr->right != NULL){
        curr = curr->right;
    }
    return curr->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: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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

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

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

typedef struct tree{
   int data;
   struct tree* left;
   struct tree* right;
}tree;

tree* createnode(int key){
   tree* nn = (tree*)malloc(sizeof(tree));
   nn->data = key;
   nn->left = NULL;
   nn->right = NULL;
   return nn;
}

tree* insert(tree* 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;
}

void range(tree* root, int L, int R, int* res, int* index){
    if (root==NULL){
        return;
    }
    if (root->data>L){
        range(root->left,L,R,res,index);
    }
    if (root->data >= L && root->data<=R){
        res[(*index)++] = root->data;
    }
    if (root->data < R){
        range(root->right,L,R,res,index);
    }
}

int main(){
    int N;
    scanf("%d",&N);
    tree* root = NULL;
    int val;
    for (int i=0;i<N;i++){
        scanf("%d",&val);
        root = insert(root,val);
    }
    int L, R;
    scanf("%d %d",&L,&R);
    int res[15];
    int index = 0;
    range(root, L,R,res,&index);
    for (int i=0;i<index;i++){
        printf("%d",res[i]);
        if (i<index-1){
            printf(" ");
        }
    }
    printf("\n");
    return 0;
```

}

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

2. Problem Statement

Arun is working on a Binary Search Tree (BST) data structure. His goal is to implement a program that reads a series of integers and inserts them into a BST. Once the integers are inserted, he needs to add a given integer value to each node in the tree and find the maximum value in the BST.

Your task is to help Arun implement this program.

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, each representing an element to be inserted into the BST.

The third line consists of an integer add, representing the value to be added to each node in the BST.

Output Format

The output prints the maximum value in the BST after adding the add value.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 25
5
Output: 30

Answer

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

```
#include <stdlib.h>

typedef struct node{
   int data;
   struct node* right;
   struct node* left;
}tree;

tree* root = NULL;

tree* createnode(int key){
   tree* nn = (tree*)malloc(sizeof(tree));
   nn->data = key;
   nn->left = NULL;
   nn->right = NULL;
   return nn;
}

tree* insert(tree* 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;
}

void add(tree* root, int n){
   if (root==NULL){
      return;
   }
   root->data += n;
   add(root->left,n);
   add(root->right,n);
}

tree* max(tree* root){
   if (root!=NULL){
```

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

int main(){
    int n, key, toadd;
    scanf("%d",&n);
    for ( int i=0;i<n;i++){
        scanf("%d",&key);
        root = insert(root,key);
    }
    scanf("%d",&toadd);
    add(root,toadd);
    tree* temp = max(root);
    printf("%d",temp->data);
    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 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>

typedef struct node{
   int data;
   struct node* left;
   struct node* right;
}tree;

tree* root = NULL;

tree* cn(int key){
   tree* nn = (tree*)malloc(sizeof(tree));
   nn->data = key;
   nn->left = NULL;
   nn->right = NULL;
   return nn;
}
```

```
tree* insert(tree* root, int key){
   if(root==NULL){
      return cn(key);
   }
   if (key<root->data){
      root->left = insert(root->left, key);
   }
   else if (key>root->data){
      root->right = insert(root->right,key);
   }
   return root;
}

tree* min(tree* root){
   if (root!=NULL){
      while(root->left!=NULL){
         root = root->left;
      }
   }
   return root;
}


tree* deleten(tree* root, int key){
   tree* temp = (tree*)malloc(sizeof(tree));
   if (key<root->data){
      root->left = deleten(root->left,key);
   }
   else if (key>root->data){
      root->right =  deleten(root->right,key);
   }
   else if (root->left && root->right){
      temp = min(root->right);
      root->data = temp->data;
      root->right = deleten(root->right,root->data);
   }
   else{
      temp = root;
      if (root->left == NULL){
         root = root->right;
      }
```

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


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

int main(){
    int n,key,todel;
    scanf("%d",&n);
    for (int i=0; i<n; i++){
        scanf("%d",&key);
        root = insert(root,key);
    }
    printf("Before deletion: ");
    inorder(root);
    printf("\n");
    scanf("%d",&todel);
    printf("After deletion: ");
    root = deleten(root,todel);
    inorder(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Kavin Sakthi R
Email: 241901044@rajalakshmi.edu.in
Roll no: 241901044
Phone: 9791145353
Branch: REC
Department: I CSE (CS) FA
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

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>
#include <math.h>

typedef struct node{
    int data;
```

```
    struct node* left = NULL;
    struct node* right = NULL;
};tree;

tree* root = NULL;

tree* createnode(int key){
    tree* nn = (tree*)malloc(sizeof(tree));
    nn->data = key;
    nn->right = NULL;
    nn->left = NULL;
    return nn;
}

tree* insert(tree* 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;
}
tree* min(tree* node){
    tree* current = node;
    while(current&&current->left!=NULL){
        current = current->left;
    }
    return current;
}
tree* deleten(tree* root,int val){
    if (root == NULL)return root;
    if (val<root->data){
        root->left = deleten(root->left,val);
    }
    else if (val>root->data){
        root->right = deleten(root->right,val);
    }
    else{
```

```
        if (root->left == NULL){
            tree* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL){
            tree* temp = root->left;
            free(root);
            return temp;
        }
        tree* temp = min(root->right);
        root->data = temp->data;
        root->right = deleten(root->right,temp->data);
    }
    return root;
}

void lot(tree* root){
    if (root == NULL)return;
    tree* queue[100];
    int front = 0, rear = 0;
    queue[rear++] = root;
    while (front<rear){
        tree* node = queue[front++];
        printf("%d ",node->data);
        if (node->left != NULL){
            queue[rear++] = node->left;
        }
        if (node->right != NULL){
            queue[rear++] = node->right;
        }
    }
}

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

```
    scanf("%d",&Y);

    for (int i=0;i<N;i++){
        root = insert(root,arr[i]);
    }
    printf("Initial BST: ");
    lot(root);
    printf("\n");

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

    root = deleten(root,Y);
    printf("BST after deleting node %d: ",Y);
    lot(root);
    printf("\n");
    return 0;
}
```

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

2. 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>

typedef struct node{
   int data;
   struct node* right;
   struct node* left;
}tree;

tree* root = NULL;

tree* createnode(int key){
   tree* newnode = (tree*)malloc(sizeof(tree));
   newnode->data = key;
   newnode->right = NULL;
   newnode->left = NULL;
   return newnode;
}

tree* insert(tree* root, int key){
   if (root == NULL){
      return createnode(key);
   }
   else if (key>root->data){
      root->right = insert(root->right,key);
```

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

void postorder(tree* root){
    if (root == NULL){
        return;
    }
    else{
        postorder(root->left);
        postorder(root->right);
        printf("%d ",root->data);
    }
}

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

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

3. 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>

typedef struct tree{
   int data;
   struct tree* right = NULL;
   struct tree* left = NULL;
}tree;

tree* root = NULL;

tree* createnode(int key){
   tree* newnode = (tree*)malloc(sizeof(tree));
   newnode->left = NULL;
   newnode->data = key;
   newnode->right = NULL;
   return newnode;
}
```

```
tree* insert(tree* root, int key){
    if (root == NULL){
        return createnode(key);
    }
    else if (key<root->data){
        root->left = insert(root->left,key);
    }
    else if (key>root->data){
        root->right = insert(root->right,key);
    }
    return root;
}

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

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

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

4. 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>

typedef struct node{
   int key;
   struct node* left;
   struct node* right;
}tree;

tree* root = NULL;
```

```
tree* createnode(int key){
    tree* nn = (tree*)malloc(sizeof(tree));
    nn->key = key;
    nn->left = NULL;
    nn->right = NULL;
    return nn;
}

tree* insert(tree* root,int key){
    if (root==NULL){
        return createnode(key);
    }
    if (key<=root->key){
        root->left =  insert(root->left,key);
    }
    else if (key>=root->key){
        root->right = insert(root->right, key);
    }
    return root;
}

tree* order(tree* root, int k1,int k2){
    if (root==NULL){
        return NULL;
    }
    root->left = order(root->left,k1,k2);
    root->right = order(root->right,k1,k2);
    if (root->key < k1){
        tree* rs = root->right;
        free(root);
        return rs;
    }
    if (root->key > k2){
        tree* ls = root->left;
        free(root);
        return ls;
    }
    return root;
}

void inorder1(tree* root){
    if (root==NULL){
```

```
        return;
    }
    inorder1(root->left);
    printf("%d ",root->key);
    inorder1(root->right);
}
void inorder2(int* count,tree* root){
        if (root==NULL||*count>=3){
            return;
        }
        inorder2(count,root->left);
        if(*count<3){
            printf("%d ",root->key);
            (*count)++;
        }

        inorder2(count,root->right);

}

int main(){
    int n,value,min,max;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d ",&value);
        root = insert(root,value);
    }
    int result = scanf("%d %d",&min,&max);
    if (result == 2){
        root = order(root,min,max);
        inorder1(root);
    }
    else{
        int count = 0;
        inorder2(&count,root);
    }
    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>

typedef struct tree{
   struct tree* left;
   int data;
   struct tree* right;
}tree;
```

```
tree* root = NULL;

tree* createnode(int key){
   tree* nn = (tree*)malloc(sizeof(tree));
   nn->left = NULL;
   nn->data = key;
   nn->right = NULL;
   return nn;
}

tree* insert(tree* root,int key){
   if (root == NULL){
      return createnode(key);
   }
   if(root->data < key){
      root->right = insert(root->right,key);
   }
   else if(root->data > key){
      root->left = insert(root->left,key);
   }
   return root;
}

int find(tree* root,int key){
   while(root!=NULL){
      if (key==root->data){
         return 1;
      }
      else if (key<root->data){
         root = root->left;
      }
      else{
         root = root->right;
      }
   }
   return 0;
}

int main(){
   int key;
   while(1){
```

```
        scanf("%d",&key);
        if (key == -1){
            break;
        }
        root = insert(root,key);
    }
    int target;
    scanf("%d",&target);
    int temp = find(root,target);
    if (temp!=0){
        printf("%d is found in the BST", target);
    }
    else{
        printf("%d is not found in the BST",target);
    }
    return 0;
}
```

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