Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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 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;
}
struct TreeNode* findMin(struct TreeNode* root)
{
    while (root->left != NULL)
    {
        root = root->left;
    }
    return root;
}
struct TreeNode* deleteNode(struct TreeNode* root, int value)
{
    if (root == NULL)
    {
        return NULL;
    }
    if (value < root->data)
    {
        root->left = deleteNode(root->left, value);
    }
    else if (value > root->data)
    {
        root->right = deleteNode(root->right, value);
    }
    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)
    {
        return;
    }
    inorderTraversal(root->left);
    printf("%d ", 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: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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;
}


void printPreorder(struct Node* root)
{
    if (root == NULL)
    {
        return;
    }
    printf("%d ", root->data);
    printPreorder(root->left);
    printPreorder(root->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: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
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);
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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 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 displayTreePostOrder(struct Node* root)
{
    if (root == NULL)
    {
        return;
    }
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ", root->data);
}
int findMinValue(struct Node* root)
{
    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: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are tasked with implementing basic operations on a queue data structure using a linked list.

You need to write a program that performs the following operations on a queue:

Enqueue Operation: Implement a function that inserts an integer element at the rear end of the queue.Print Front and Rear: Implement a function that prints the front and rear elements of the queue. Dequeue Operation: Implement a function that removes the front element from the queue.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the queue.

The second line consists of N space-separated integers, representing the queue elements.

Output Format

The first line prints "Front: X, Rear: Y" where X is the front and Y is the rear elements of the queue.

The second line prints the message indicating that the dequeue operation (front element removed) is performed: "Performing Dequeue Operation:".

The last line prints "Front: M, Rear: N" where M is the front and N is the rear elements after the dequeue operation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
12 56 87 23 45

Output: Front: 12, Rear: 45
Performing Dequeue Operation:
Front: 56, Rear: 45

Answer

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

struct Node {
    int data;
    struct Node* next;
};

struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int d)
{
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = d;
```

```
    newNode->next = NULL;
    if (rear == NULL)
    {
        front = rear = newNode;
    }
    else
    {
        rear->next = newNode;
        rear = newNode;
    }
}

void printFrontRear()
{
    if (front != NULL && rear != NULL)
    {
        printf("Front: %d, Rear: %d\n", front->data, rear->data);
    }
}

void dequeue()
{
    if (front != NULL)
    {
        struct Node* temp = front;
        front = front->next;
        if (front == NULL)
        {
            rear = NULL;
        }
        free(temp);
    }
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
    printf("Performing Dequeue Operation:\n");
```

```
    dequeue();
    printFrontRear();
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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;
}root;

struct node *create(int key)
{
   struct node *nnode=(struct node*)malloc(sizeof(struct node));
   nnode->data=key;
   nnode->left=NULL;
   nnode->right=NULL;
   return nnode;
}

struct node *insert(struct node *root,int key)
{
   if(root==NULL)
   return create(key);
```

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

void display(struct node *root,int min,int max)
{
    if(root!=NULL)
    {
        display(root->left,min,max);
        if(root->data>=min && root->data<=max)
        printf("%d ",root->data);
        display(root->right,min,max);
    }
}
int main()
{
    int n;
    struct node*root=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int data;
        scanf("%d",&data);
        root=insert(root,data);
    }
    int max,min;
    scanf("%d %d",&min,&max);
    display(root,min,max);
}
```

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>

struct node
{
   int data;
   struct node *left;
   struct node *right;
}root;

struct node *create(int key)
{
   struct node *nnode=(struct node*)malloc(sizeof(struct node));
   nnode->data=key;
   nnode->left=nnode->right=NULL;
```

```
    return nnode;
}

struct node *insert(struct node *root,int key)
{
    if(root==NULL)
    return create(key);
    else if(root->data>key)
    root->left=insert(root->left,key);
    else
    root->right=insert(root->right,key);
    return root;
}

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

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

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

3. 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>

struct node
{
   int data;
   struct node *left;
   struct node *right;
}root;

struct node *create(int key)
{
   struct node* nnode=(struct node*)malloc(sizeof(struct node));
   nnode->data=key;
   nnode->left=nnode->right=NULL;
   return nnode;
}

struct node *insert(struct node* root,int key)
{
   if(root==NULL)
   {
      return create(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;
```

```
}
struct node *findmin(struct node *root)
{
   while(root && root->left!=NULL)
   {
      root=root->left;
   }
   return root;
}

struct node *del(struct node *root,int key)
{
   if(root==NULL)
   return NULL;
   if(key<root->data)
   {
      root->left=del(root->left,key);
   }
   else if(key>root->data)
   {
      root->right=del(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;
      }
      else
      {
         struct node *temp=findmin(root->right);
         root->data=temp->data;
         root->right=del(root->right,temp->data);
```

```
        }
    }
    return root;
}

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

int main()
{
    int n;
    scanf("%d",&n);
    int key;
    struct node *root=NULL;
    for(int i=0;i<n;i++)
    {
        scanf("%d",&key);
        root=insert(root,key);
    }
    printf("Initial BST:");
    levelorder(root);
    int x,y;
    scanf("%d %d",&x,&y);
    root=insert(root,x);
    printf("\nBST after inserting a new node %d: ",x);
    levelorder(root);
    root=del(root,y);
```

```
    printf("\nBST after deleting node %d:",y);
    levelorder(root);

}
```

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

4. 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>

struct node
{
   int data;
   struct node*left;
   struct node*right;
}root;

struct node *create(int key)
{
   struct node*nnode=(struct node*)malloc(sizeof(struct node));
   nnode->data=key;
   nnode->left=nnode->right=NULL;
   return nnode;
}

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

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

```
int main()
{
    struct node *root=NULL;
    int n;
    do
    {
        scanf("%d",&n);
        root=insert(root,n);
    }while(n!=-1);
    int val;
    scanf("%d",&val);
    search(root,val);
    if(search(root,val))
    printf("%d is found in the BST",val);
    else
    printf("%d is not found in the BST",val);
}
```

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

5. 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;
}root;

struct node *create(int key)
{
   struct node *nnode=(struct node*)malloc(sizeof(struct node));
   nnode->data=key;
   nnode->left=nnode->right=NULL;
   return nnode;
}

struct node *insert(struct node *root,int key)
{
   if(root==NULL){
      return create(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 postorder(struct node*root)
{
    if(root!=NULL){
        postorder(root->left);
        postorder(root->right);
        printf("%d ",root->data);
    }
}

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

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