Rajalakshmi Engineering College

Name: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l 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 : 15

Section 1 : MCQ

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

Answer

14

Status : Correct   *Marks : 1/1*

2. 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*

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

Answer

1, 2, 4, 13, 14, 18

Status : Correct   *Marks : 1/1*

4. 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*

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

Answer

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

Status : Correct   *Marks : 1/1*

6. 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*

7. 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*

8.   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*

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

Answer

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

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

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

Answer

1, 4, 2, 18, 14, 13

Status : Correct *Marks : 1/1*

13. 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*

14. 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*

15. 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*

Rajalakshmi Engineering College

Name: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l 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

```c
#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)
```

```c
        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) {
    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 == 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);
    }
```

```c
        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: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l 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

```c
#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) {
    //Type your code here
    if(root==NULL){
        return createNode(value);
    }
    if(value>root->data){
        root->left=insert(root->left,value);
    }
    else if(value<root->data){
        root->right=insert(root->right,value);
    }
```

```c
        return root;
    }
void printPreorder(struct Node* node) {
    //Type your code here
    if(node==NULL){
        return;
    }
    printf("%d",node->data);
    printPreorder(node->right);
    printPreorder(node->left);
}

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: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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) {
    //Type your code here
    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) {
    //Type your code here
    if(root==NULL||root->data==value){
        return root;
```

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

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l 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

```c
#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) {
    //Type your code here
    if(root==NULL){
        return createNode(key);
    }
    if(key<root->data){
        root->left=insert(root->left,key);
    }
    else{
        root->right=insert(root->right,key);
    }
    return root;
```

```c
    }
    int findMax(struct TreeNode* root) {
        //Type
        if(root==NULL){
            return root->data;
        }
        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: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 47.5

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int key;
    struct Node* left;
```

```c
    struct Node* right;
} Node;
typedef struct Queue {
    Node** arr;
    int front, rear, size;
} Queue;
Node* createNode(int key) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->key = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}
int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}
Node* buildBalancedBST(int arr[], int start, int end) {
    if (start > end) return NULL;
    int mid = (start + end) / 2;
    Node* root = createNode(arr[mid]);
    root->left = buildBalancedBST(arr, start, mid - 1);
    root->right = buildBalancedBST(arr, mid + 1, end);
    return root;
}
Node* insertNode(Node* root, int key) {
    if (root == NULL) return createNode(key);
    if (key < root->key)
        root->left = insertNode(root->left, key);
    else if (key > root->key)
        root->right = insertNode(root->right, key);
    return root;
}
Node* minValueNode(Node* node) {
    Node* current = node;
    while (current && current->left != NULL)
        current = current->left;
    return current;
}
Node* deleteNode(Node* root, int key) {
    if (root == NULL) return root;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
```

```c
            root->right = deleteNode(root->right, key);
        else {
            if (root->left == NULL) {
                Node* temp = root->right;
                free(root);
                return temp;
            } else if (root->right == NULL) {
                Node* temp = root->left;
                free(root);
                return temp;
            }
            Node* temp = minValueNode(root->right);
            root->key = temp->key;
            root->right = deleteNode(root->right, temp->key);
        }
    return root;
}
Queue* createQueue(int size) {
Queue* q = (Queue*)malloc(sizeof(Queue));
q->arr = (Node**)malloc(sizeof(Node*) * size);
q->front = q->rear = 0;
q->size = size;
return q;
}
void enqueue(Queue* q, Node* node) {
q->arr[q->rear++] = node;
}
Node* dequeue(Queue* q) {
return q->arr[q->front++];
}
int isEmpty(Queue* q) {
return q->front == q->rear;
}
void levelOrder(Node* root) {
if (root == NULL) return;
Queue* q = createQueue(100);
enqueue(q, root);
while (!isEmpty(q)) {
Node* curr = dequeue(q);
printf("%d ", curr->key);
if (curr->left) enqueue(q, curr->left);
if (curr->right) enqueue(q, curr->right);
```

```
        }
        free(q->arr);
        free(q);
    }

    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);

        Node* root = NULL;
        for (int i = 0; i < N; ++i) {
            root = insertNode(root, arr[i]);
        }

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

        root = insertNode(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*

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
```

```c
}
Node* insert(Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}
void postOrder(Node* root) {
    if (root == NULL) return;
    postOrder(root->left);
    postOrder(root->right);
    printf("%d ", root->data);
}

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

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

    postOrder(root);
    printf("\n");
    return 0;
}
```

Status : Correct   *Marks : 10/10*

3. 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

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}
Node* insert(Node* root, int data) {
    if (root == NULL)
```

```
    return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}
int search(Node* root, int target) {
    if (root == NULL)
        return 0;
    if (root->data == target)
        return 1;
    else if (target < root->data)
        return search(root->left, target);
    else
        return search(root->right, target);
}

int main() {
    Node* root = NULL;
    int value;
    while (scanf("%d", &value) && value != -1) {
        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*

4. 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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

int index = 0;


typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
```

```c
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}
Node* constructBST(int preorder[], int* indexPtr, int key, int min, int max, int n) {
    if (*indexPtr >= n) return NULL;

    Node* root = NULL;
    if (key > min && key < max) {
        root = createNode(key);
        (*indexPtr)++;

        if (*indexPtr < n) {
            root->left = constructBST(preorder, indexPtr, preorder[*indexPtr], min, key, n);
        }
        if (*indexPtr < n) {
            root->right = constructBST(preorder, indexPtr, preorder[*indexPtr], key, max, n);
        }
    }

    return root;
}

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

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

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

    int currentIndex = 0;
```

```
    Node* root = constructBST(preorder, &currentIndex, preorder[0], INT_MIN,
INT_MAX, n);

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

    return 0;
}
```

Status : Correct   *Marks : 10/10*

5. 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

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

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

Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

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

Node* trimBST(Node* root, int min, int max) {
    if (root == NULL)
        return NULL;

    root->left = trimBST(root->left, min, max);
    root->right = trimBST(root->right, min, max);
```

```c
    if (root->data < min) {
        Node* rightSubtree = root->right;
        free(root);
        return rightSubtree;
    }
    if (root->data > max) {
        Node* leftSubtree = root->left;
        free(root);
        return leftSubtree;
    }
    return root;
}

void inorderTraversal(Node* root, int* isFirst) {
    if (root == NULL) return;
    inorderTraversal(root->left, isFirst);
    if (*isFirst) {
        printf("%d", root->data);
        *isFirst = 0;
    } else {
        printf(" %d", root->data);
    }
    inorderTraversal(root->right, isFirst);
}

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

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

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

```c
    }

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

    root = trimBST(root, min, max);

    int isFirst = 1;
    inorderTraversal(root, &isFirst);
    printf("\n");

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

Status : Partially correct   *Marks : 7.5/10*

Rajalakshmi Engineering College

Name: Barath Raj M
Email: 241901015@rajalakshmi.edu.in
Roll no: 241901015
Phone: 7358484658
Branch: REC
Department: l 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

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

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}
Node* insert(Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}
void findKthLargest(Node* root, int k, int* count, int* result) {
    if (root == NULL || *count >= k) return;
```

```c
    findKthLargest(root->right, k, count, result);

    (*count)++;
    if (*count == k) {
        *result = root->data;
        return;
    }

    findKthLargest(root->left, k, count, result);
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    Node* root = NULL;

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

    if (k > n || k <= 0) {
        printf("Invalid value of k\n");
    } else {
        int count = 0, result = -1;
        findKthLargest(root, k, &count, &result);
        printf("%d\n", result);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

2. Problem Statement

Dhruv is working on a project where he needs to implement a Binary

Search Tree (BST) data structure and perform various operations on it.

He wants to create a program that allows him to build a BST, traverse it in different orders (inorder, preorder, postorder), and exit the program when needed.

Help Dhruv by designing a program that fulfils his requirements.

Input Format

The first input consists of the choice.

If the choice is 1, enter the number of elements N and the elements inserted into the tree, separated by a space in a new line.

If the choice is 2, print the in-order traversal.

If the choice is 3, print the pre-order traversal.

If the choice is 4, print the post-order traversal.

If the choice is 5, exit.

Output Format

The output prints the results based on the choice.

For choice 1, print "BST with N nodes is ready to use" where N is the number of nodes inserted.

For choice 2, print the in-order traversal of the BST.

For choice 3, print the pre-order traversal of the BST.

For choice 4, print the post-order traversal of the BST.

For choice 5, the program exits.

If the choice is greater than 5, print "Wrong choice".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
5
12 78 96 34 55
2
3
4
5

Output: BST with 5 nodes is ready to use
BST Traversal in INORDER
12 34 55 78 96
BST Traversal in PREORDER
12 78 34 55 96
BST Traversal in POSTORDER
55 34 96 78 12

Answer

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
int data;
struct Node* left;
struct Node* right;
} Node;
Node* createNode(int data) {
Node* newNode = (Node*) malloc(sizeof(Node));
newNode->data = data;
newNode->left = newNode->right = NULL;
return newNode;
}
Node* insert(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 inorderTraversal(Node* root) {
```

```c
    if (root == NULL) return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorderTraversal(root->right);
}
void preorderTraversal(Node* root) {
if (root == NULL) return;
printf("%d ", root->data);
preorderTraversal(root->left);
preorderTraversal(root->right);
}
void postorderTraversal(Node* root) {
    if (root == NULL) return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ", root->data);
}
void freeTree(Node* root) {
    if (root == NULL) return;
    freeTree(root->left);
    freeTree(root->right);
    free(root);
}
int main() {
    int choice;
    Node* root = NULL;

    while (1) {
        if (scanf("%d", &choice) != 1) break;

        switch (choice) {
            case 1: {
                int N;
                if (scanf("%d", &N) != 1) break;
                freeTree(root);
                root = NULL;

                for (int i = 0; i < N; i++) {
                    int val;
                    if (scanf("%d", &val) != 1) break;
                    root = insert(root, val);
                }
```

```
            printf("BST with %d nodes is ready to use\n", N);
            break;
        }

        case 2:
            printf("BST Traversal in INORDER\n");
            inorderTraversal(root);
            printf("\n");
            break;

        case 3:
            printf("BST Traversal in PREORDER\n");
            preorderTraversal(root);
            printf("\n");
            break;

        case 4:
            printf("BST Traversal in POSTORDER\n");
            postorderTraversal(root);
            printf("\n");
            break;

        case 5:
            return 0;

        default:
            printf("Wrong choice\n");
            break;
        }
    }

    return 0;
}
```

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

3. 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

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}
```

```c
Node* insert(Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}
int search(Node* root, int key) {
    if (root == NULL)
        return 0;
    if (root->data == key)
        return 1;
    else if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}

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

    Node* root = NULL;
    int val;

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

    int key;
    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;
}
```