# Rajalakshmi Engineering College

Name: Nikhil Vinayak P
Email: 240701359@rajalakshmi.edu.in
Roll no: 240701359
Phone: 9884558531
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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

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***

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct BSTNode {
    int value;
    struct BSTNode* left;
    struct BSTNode* right;
};
struct BSTNode* createNode(int key) {
    struct BSTNode* newNode=(struct BSTNode*)malloc(sizeof(struct
BSTNode));
    newNode->value=key;
    newNode->left=newNode->right=NULL;
    return newNode;
}
struct BSTNode* insert(struct BSTNode* root,int key) {
    if (root==NULL) {
        return createNode(key);
    }
    if (key<root->value) {
        root->left=insert(root->left,key);
    }
    else {
        root->right=insert(root->right,key);
    }
```

```
    return root;
}
int search(struct BSTNode* root,int key) {
    if (root==NULL) {
        return 0;
    }
    if (root->value==key) {
        return 1;
    }
    if (key<root->value) {
        return search(root->left,key);
    }
    else {
        return search(root->right,key);
    }
}
int main() {
    int n,key;
    scanf("%d", &n);
    struct BSTNode* root=NULL;
    for (int i=0; i<n; i++) {
        int value;
        scanf("%d", &value);
        root=insert(root,value);
    }
    scanf("%d", &key);
    if (search(root,key)) {
        printf("The key %d is found in the binary search tree\n", key);
    }
    else {
        printf("The key %d is not found in the binary search tree\n", key);
    }
    return 0;
}
```

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

2. 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***

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct Node {
```

```
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* createNode(int data) {
    struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
    newNode->data=data;
    newNode->left=newNode->right=NULL;
    return newNode;
}
struct Node* insert(struct Node* root,int data) {
    if (!root)
        return createNode(data);
    if (data<root->data)
        root->left=insert(root->left,data);
    else
        root->right=insert(root->right,data);
    return root;
}
void inorder(struct Node* root) {
    if (root) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
struct Node* findMin(struct Node* root) {
    while (root && root->left)
        root=root->left;
    return root;
}
struct Node* deleteNode(struct Node* root,int key) {
    if (!root) 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) {
            struct Node* temp=root->right;
            free(root);
            return temp;
```

```
        }
        else if (!root->right) {
            struct Node* temp=root->left;
            free(root);
            return temp;
        }
        struct Node* temp=findMin(root->right);
        root->data=temp->data;
        root->right=deleteNode(root->right,temp->data);
    }
    return root;
}
int search(struct Node* root,int key) {
    if (!root) return 0;
    if (root->data==key) return 1;
    if (key<root->data)
        return search(root->left,key);
    return search(root->right,key);
}
int main() {
    int n,x;
    scanf("%d", &n);
    struct Node* root=NULL;
    for (int i=0; i<n; i++) {
        int val;
        scanf("%d", &val);
        root=insert(root,val);
    }
    scanf("%d", &x);
    printf("Before deletion: ");
    inorder(root);
    printf("\n");
    if (search(root,x))
        root=deleteNode(root,x);
    printf("After deletion: ");
    inorder(root);
    printf("\n");
    return 0;
}
```

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

3. 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***

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *left, *right;
};
struct Node* createNode(int data) {
```

```
    struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
    newNode->data=data;
    newNode->left=newNode->right=NULL;
    return newNode;
}
struct Node* insert(struct Node* root,int data) {
    if (root==NULL) return createNode(data);
    if (data<root->data)
        root->left=insert(root->left,data);
    else
        root->right=insert(root->right,data);
    return root;
}
void rangeSearch(struct Node* root,int L,int R) {
    if (root==NULL) return;
    if (root->data>L)
        rangeSearch(root->left,L,R);
    if (root->data>=L && root->data<=R)
        printf("%d ", root->data);
    if (root->data<R)
        rangeSearch(root->right,L,R);
}
int main() {
    int n,L,R;
    scanf("%d", &n);
    struct Node* root=NULL;
    for (int i=0; i<n; i++) {
        int val;
        scanf("%d", &val);
        root=insert(root,val);
    }
    scanf("%d %d", &L, &R);
    rangeSearch(root,L,R);
    printf("\n");
    return 0;
}
```

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