# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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. 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***

2. 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***

3. 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***

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

***Answer***

1, 2, 4, 13, 14, 18

***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 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 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**

8. 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**

9. 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**

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

***Answer***

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

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

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

***Answer***

14

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

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. 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***

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

***Answer***

13, 2, 1, 4, 14, 18

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

# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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*

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

struct TreeNode* deleteNode(struct TreeNode* root, int key) { if
    (root == NULL) return root;
    if (key < root->data) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->data) {
        root->right = deleteNode(root->right, key);
    } else {
        if (root->left == NULL) {
            struct TreeNode* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct TreeNode* temp = root->left;
            free(root);
            return temp;
        }
        struct TreeNode* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

void inorderTraversal(struct TreeNode* root) { if
    (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", 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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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*

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

```
void printPreorder(struct Node* node) { if
   (node == NULL)
      return;
   printf("%d ", node->data);
   printPreorder(node->left);
   printPreorder(node->right);
}

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

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

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

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

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

# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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 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*

```
Node* insertNode(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;
}

Node* searchNode(Node* root, int value) { if
   (root == NULL || root->data == value) {
      return root;
   }
```

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

    return searchNode(root->right, value);
}
```

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

# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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 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 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 displayTreePostOrder(struct Node* root) { if
    (root == NULL) {
        return;
    }
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ", root->data);
}

int findMinValue(struct Node* root) { if
    (root == NULL) {
        return 1000000;
    }

    int leftMin = findMinValue(root->left); int
    rightMin = findMinValue(root->right);

    int min = root->data; if
    (leftMin < min) {
        min = leftMin;
    }
    if (rightMin < min) {
        min = rightMin;
    }

    return min;
}

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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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*

```
#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;
}

int findMax(struct TreeNode* root) { if
    (root == NULL) return -1;
    while (root->right != NULL) {
        root = root->right;
```

```
    }
    return root->data;
}

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

    struct TreeNode* root = NULL;

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

    int maxVal = findMax(root); if
    (maxVal != -1) {
        printf("%d", maxVal);
    }

    return 0;
}
```

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

# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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

John is building a system to store and manage integers using a binary search tree (BST). He needs to add a feature that allows users to search for a specific integer key in the BST using recursion.

Implement functions to create the BST and perform a recursive search for an integer.

***Input Format***

The first line of input consists of an integer representing, the number of nodes.

The second line consists of integers representing, the values of nodes, separated by space.

The third line consists of an integer representing, the key to be searched.

### *Output Format*

The output prints whether the given key is present in the binary search tree or not.

Refer to the sample output for the exact format.

### *Sample Test Case*

Input: 7
10 5 15 3 7 12 20
12
Output: The key 12 is found in the binary search tree

### *Answer*

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

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

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

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

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

    return root;
}

int searchKey(struct Node* root, int key) { if
    (root == NULL || root->data == key) {
        return root != NULL;
    }

    if (key < root->data) {
        return searchKey(root->left, key);
    } else {
        return searchKey(root->right, key);
    }
}

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

    scanf("%d", &numNodes);

    for (int i = 0; i < numNodes; i++) { scanf("%d",
        &value);
        root = insertNode(root, value);
    }

    scanf("%d", &key);

    int found = searchKey(root, key); if
    (found) {
        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***

```
#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 = NULL;
   newNode->right = NULL;
   return newNode;
}

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

struct Node* getMinimumKey(struct Node* curr) {
   while (curr->left != NULL) {
      curr = curr->left;
   }
   return curr;
}

struct Node* insert(struct Node* root, int key) { 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;
}

struct Node* deleteNode(struct Node* root, int key) { struct
    Node* parent = NULL;
    struct Node* curr = root;

    while (curr != NULL && curr->data != key) {
        parent = curr;
        if (key < curr->data) {
            curr = curr->left;
        } else {
            curr = curr->right;
        }
    }

    if (curr == NULL) {
        return root;
    }

    if (curr->left == NULL && curr->right == NULL) {
        if (curr != root) {
            if (parent->left == curr) {
                parent->left = NULL;
            } else {
                parent->right = NULL;
            }
            free(curr);
        } else {
            root = NULL;
        }
    } else if (curr->left && curr->right) {
        struct Node* successor = getMinimumKey(curr->right); int val
        = successor->data;
        root = deleteNode(root, successor->data); curr-
        >data = val;
    } else {
```

```
        struct Node* child = (curr->left != NULL) ? curr->left : curr->right; if
        (curr != root) {
            if (parent->left == curr) {
                parent->left = child;
            } else {
                parent->right = child;
            }
            free(curr);
        } else {
            root = child;
        }
    }
    return root;
}

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

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

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

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

    root = deleteNode(root, delKey);

    printf("\nAfter deletion: ");
    inorder(root);

    return 0;
}
```

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

3. 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*

```
#include    <stdio.h>
#include<stdlib.h>
struct tnode
{
   int data;
   struct tnode *right;
   struct tnode *left;
};

struct tnode *CreateBST(struct tnode *root, int item)
{
   if(root == NULL)
   {
      root = (struct tnode *)malloc(sizeof(struct tnode)); root-
      >left = root->right = NULL;
      root->data = item;
      return root;
   }
   else
   {
```

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

        return(root);
    }
}

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

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

void Postorder(struct tnode *root)
{
    if( root != NULL)
    {
        Postorder(root->left);
        Postorder(root->right);
        printf("%d ",root->data);
    }
}
int main()
{
    struct tnode *root = NULL;
    int choice, item, n, i;
```

```
    do
    {
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            root = NULL;
            scanf("%d",&n);
            for(i = 1; i <= n; i++)
            {
                scanf("%d",&item);
                root = CreateBST(root,item);
            }
            printf("BST with %d nodes is ready to use\n", n);
            break;
        case 2:
            printf("BST Traversal in INORDER\n");
            Inorder(root);
            printf("\n");
            break;
        case 3:
            printf("BST Traversal in PREORDER\n");
            Preorder(root);
            printf("\n");
            break;
        case 4:
            printf("BST Traversal in POSTORDER\n");
            Postorder(root);
            printf("\n");
            break;
        case 5:
            exit(0);
            break;
        default:
            printf("Wrong choice\n");
            break;
        }
    } while(1);
    return 0;
}
```

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

# Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
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_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1. 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>
#include <stdbool.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 == 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;
}

bool search(struct Node* root, int target) { if
    (root == NULL) {
        return false;
    }

    if (root->data == target) {
        return true;
    }

    if (target < root->data) {
        return search(root->left, target);
    } else {
        return search(root->right, target);
    }
}

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

    int num;
    while (1) {
        scanf("%d", &num);
        if (num == -1) {
            break;
        }
        root = insert(root, num);
    }

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

    if (search(root, target)) {
        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**

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>
```

```
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 == 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 displayTree(struct Node* root) { if
    (root == NULL) {
        return;
    }
    displayTree(root->left);
    displayTree(root->right);
    printf("%d ", 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);
    }

    displayTree(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>

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

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

struct node* constructTreeUtil(int pre[], int* preIndex, int low, int high, int size) { if
   (*preIndex >= size || low > high)
      return NULL;

   struct node* root = newNode(pre[*preIndex]);
   *preIndex = *preIndex + 1;

   if (low == high)
      return root;

   int i;
   for (i = low; i <= high; ++i)
      if (pre[i] > root->data)
         break;

   root->left = constructTreeUtil(pre, preIndex, *preIndex, i - 1, size); root-
   >right = constructTreeUtil(pre, preIndex, i, high, size);

   return root;
}

struct node* constructTree(int pre[], int size) { int
   preIndex = 0;
```

```
    return constructTreeUtil(pre, &preIndex, 0, size - 1, size);
}

void printInorder(struct node* node) { if
    (node == NULL)
        return;

    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}

int main() {
    int i, size;
    scanf("%d", &size);
    int pre[size];

    for (i = 0; i < size; i++) {
        scanf("%d", &pre[i]);
    }

    struct node* root = constructTree(pre, size);
    printInorder(root);
    printf("\n");

    return 0;
}
```

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

4. Problem Statement

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

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

InsertionDeletion

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

### *Input Format*

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

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

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

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

### *Output Format*

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

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

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

Refer to the sample output for formatting specifications.

### *Sample Test Case*

Input: 5
25 14 56 28 12
34
12

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

### *Answer*

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

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

struct TreeNode {
    int val;
    struct TreeNode *left, *right;
};
struct TreeNode* createNode(int x) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    newNode->val = x;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct TreeNode* insert(struct TreeNode* root, int val) { if
    (!root) return createNode(val);
    if (val < root->val) root->left = insert(root->left, val); else
    root->right = insert(root->right, val);
    return root;
}
struct TreeNode* deleteNode(struct TreeNode* root, int key) { if
    (!root) return root;
    if (key < root->val) root->left = deleteNode(root->left, key);
    else if (key > root->val) root->right = deleteNode(root->right, key); else
    {
        if (!root->left) {
            struct TreeNode* temp = root->right;
            free(root);
            return temp;
        } else if (!root->right) {
            struct TreeNode* temp = root->left;
            free(root);
            return temp;
        }
        struct TreeNode* temp = root->right;
        while (temp && temp->left) temp = temp->left;
        root->val = temp->val;
        root->right = deleteNode(root->right, temp->val);
    }
    return root;
}
```

```
void levelOrderTraversal(struct TreeNode* root) { if
    (!root) return;
    struct TreeNode* queue[1000]; int
    front = 0, rear = 0; queue[rear++]
    = root;
    while (front < rear) {
        struct TreeNode* current = queue[front++]; printf("%d ",
        current->val);
        if (current->left) queue[rear++] = current->left;
        if (current->right) queue[rear++] = current->right;
    }
}

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

    printf("Initial BST: ");
    levelOrderTraversal(root);
    printf("\n");
    scanf("%d", &x);
    root = insert(root, x);
    printf("BST after inserting a new node %d: ", x);
    levelOrderTraversal(root);
    printf("\n");
    scanf("%d", &y);
    root = deleteNode(root, y);
    printf("BST after deleting node %d: ", y);
    levelOrderTraversal(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*

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

struct node {
    int key;
```

```
    struct node* left;
    struct node* right;
};

struct node* newNode(int num) {
    struct node* temp = (struct node*)malloc(sizeof(struct node)); temp-
    >key = num;
    temp->left = temp->right = NULL; return
    temp;
}

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

struct node* removeOutsideRange(struct node* root, int min, int max) { if (root
    == NULL)
        return NULL;
    root->left = removeOutsideRange(root->left, min, max); root-
    >right = removeOutsideRange(root->right, min, max); if (root-
    >key < min) {
        struct node* rChild = root->right; free(root);
        return rChild;
    }
    if (root->key > max) {
        struct node* lChild = root->left;
        free(root);
        return lChild;
    }
    return root;
}

void inorderTraversal(struct node* root) { if
    (root) {
        inorderTraversal(root->left);
```

```
        printf("%d ", root->key);
        inorderTraversal(root->right);
    }
}

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

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