Practical – 8 Implementation of Tree and Searching

1. **Write a program to do the following operations.**
    - **Create a Binary Tree by collecting information from users.**
    - **Create a Binary Search Tree by collecting information from users.**
    - **Traverse the created trees using ○ preorder ○ postorder ○ inorder ○ levelorder**
    - **Search Element in Binary Search Tree**
    - **Find Internal Nodes, External Nodes, Total Nodes and Height of Tree**

Code :

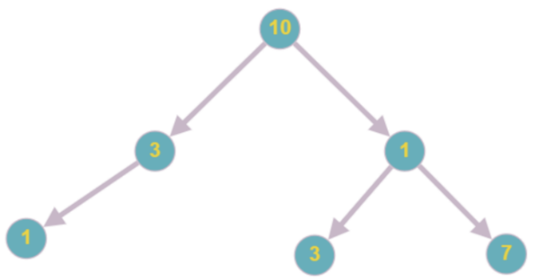
```
#include <stdio.h>
#include <stdlib.h.>
struct node
{
    int data;
    struct node *left;
    struct node *right;
};

// Create Node
struct node* create(int value) {
    struct node* newNode = malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
struct node* insertInBinaryTree(struct node* node, int data)
{
    if (node == NULL)
        return create(data);

    if (data < node->data)
        node->left = insertInBinaryTree(node->left, data);
    else if (data > node->data)
        node->right = insertInBinaryTree(node->right, data);

    return node;
}

struct node* searchInBST(struct node* root, int key){
    if(root==NULL) return NULL;

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

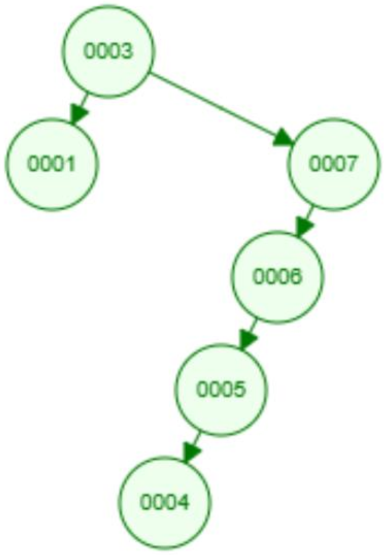
```
struct node* createBinerytree(struct node* root)
{
      int n;
      int f = 0;
      int r = 0;
      n = 100;

      struct node* q[n];

      struct node *new,*temp;
      int data;
      int ri;
      int le;

      printf("\nEnter root node : ");
      scanf("%d", &data);

      new = create(data);
      root = new;

      q[r++] = new;

      do
      {
            temp = q[f++];

            printf("\nEnter Child of %d\n", temp->data);
            printf("Right child :");
            scanf("%d", &ri);
            printf("Left child :");
            scanf("%d", &le);

            if (ri > 0)
            {
                  new = create(ri);

                  temp->right = new;
                  q[r++] = new;
            }

            if (le > 0)
            {
                  new = create(le);

                  temp->left = new;
                  q[r++] = new;
            }
      } while (f != r);
      return root;
}
```

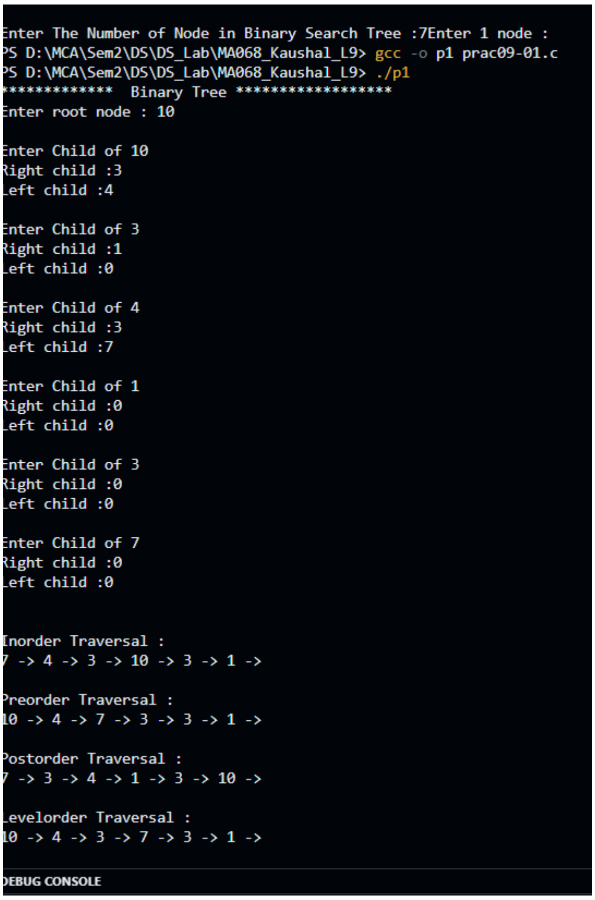
```
void inOrder(struct node* root){

    if(root==NULL)return;

    inOrder(root->left);
    printf("%d -> ",root->data);
    inOrder(root->right);

}
void preOrder(struct node* root){

    if(root==NULL)return;

    printf("%d -> ",root->data);
    preOrder(root->left);
    preOrder(root->right);

}
void postOrder(struct node* root){

    if(root==NULL)return;

    postOrder(root->left);
    postOrder(root->right);
    printf("%d -> ",root->data);

}

void levelOrder(struct node* root) {
    if (root == NULL) return;
    struct node* queue[1000];
    int front = 0, rear = 0;
    queue[rear++] = root;

    while (front < rear) {
        struct node* node = queue[front++];
        printf("%d -> ", node->data);
        if (node->left != NULL)
            queue[rear++] = node->left;
        if (node->right != NULL)
            queue[rear++] = node->right;
    }
}

int countInternalNodes(struct node* root) {
    if (root == NULL)
        return 0;
    if (root->left == NULL && root->right == NULL)
        return 0;

    return countInternalNodes(root->left) + countInternalNodes(root-
>right)+1;
}
```

```
int countExternalNodes(struct node* root) {
    if (root == NULL)
        return 0;
    if (root->left == NULL && root->right == NULL)
        return 1;
    return countExternalNodes(root->left) + countExternalNodes(root->right);
}

int countNodes(struct node* root) {
    if (root == NULL)
        return 0;

    return 1 + countNodes(root->left) + countNodes(root->right);
}

int height(struct node* root) {
    if (root == NULL)
        return -1;
    int leftHeight = height(root->left);
    int rightHeight = height(root->right);
    return 1 + (leftHeight > rightHeight ? leftHeight : rightHeight);
}


int main(){

    // Create Binary Tree
    printf("*************  Binary Tree ******************");
    struct node* root;
    root =createBinerytree(root);


    printf("\n\nInorder Traversal :\n");
    inOrder(root);

    printf("\n\nPreorder Traversal :\n");
    preOrder(root);

    printf("\n\nPostorder Traversal :\n");
    postOrder(root);

    printf("\n\nLevelorder Traversal :\n");
    levelOrder(root);

    // Create BST
    printf("\n\n*************  Binary Search Tree ******************\n");
    int Bnodes,data;
    printf("\nEnter The Number of Node in Binary Search Tree :");
    scanf("%d",&Bnodes);

    struct node* broot = NULL;
    printf("Enter 1 node :");
    scanf("%d",&data);
    broot =insertInBinaryTree(broot,data);

    for(int i=2; i<=Bnodes; i++){
        printf("\nEnter %d node : ",i);
        scanf("%d",&data);
        insertInBinaryTree(broot,data);
    }

    // Search in BST
    int sdata;
    printf("\n\nEnter element for Search in BST :");
    scanf("%d",&sdata);

    struct node* n = searchInBST(broot, sdata);
    if(n!=NULL){
        printf("Found: %d", n->data);
    }
    else{
        printf("Element not found");
    }

    printf("\n\nInorder Traversal :\n");
    inOrder(broot);

    printf("\n\nPreorder Traversal :\n");
    preOrder(broot);

    printf("\n\nPostorder Traversal :\n");
    postOrder(broot);

    printf("\n\nLevelorder Traversal :\n");
    levelOrder(broot);

    printf("\n\nInternal Node in BST : %d\n",countInternalNodes(broot));
    printf("\nExternal Node in BST : %d\n",countExternalNodes(broot));
    printf("\nTotal Node in BST : %d\n",countNodes(broot));
    printf("\nHeight Node in BST : %d\n",height(broot));

    return 0;

}
```

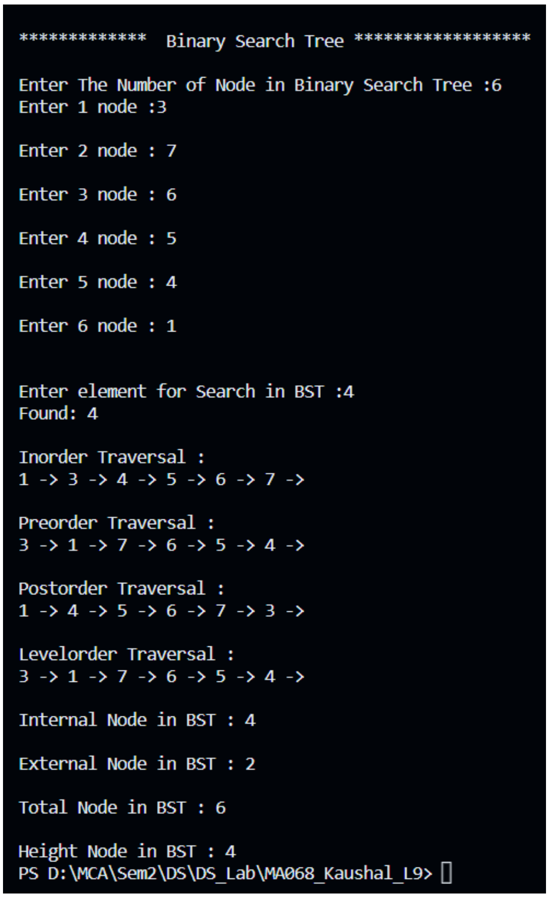
Output :







```
Enter The Number of Node in Binary Search Tree :7Enter 1 node :
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9> gcc -o p1 prac09-01.c
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9> ./p1
*************  Binary Tree ******************
Enter root node : 10

Enter Child of 10
Right child :3
Left child :4

Enter Child of 3
Right child :1
Left child :0

Enter Child of 4
Right child :3
Left child :7

Enter Child of 1
Right child :0
Left child :0

Enter Child of 3
Right child :0
Left child :0

Enter Child of 7
Right child :0
Left child :0


Inorder Traversal :
7 -> 4 -> 3 -> 10 -> 3 -> 1 ->

Preorder Traversal :
10 -> 4 -> 7 -> 3 -> 3 -> 1 ->

Postorder Traversal :
7 -> 3 -> 4 -> 1 -> 3 -> 10 ->

Levelorder Traversal :
10 -> 4 -> 3 -> 7 -> 3 -> 1 ->
```



```
*************  Binary Search Tree ******************

Enter The Number of Node in Binary Search Tree :6
Enter 1 node :3

Enter 2 node : 7

Enter 3 node : 6

Enter 4 node : 5

Enter 5 node : 4

Enter 6 node : 1


Enter element for Search in BST :4
Found: 4

Inorder Traversal :
1 -> 3 -> 4 -> 5 -> 6 -> 7 ->

Preorder Traversal :
3 -> 1 -> 7 -> 6 -> 5 -> 4 ->

Postorder Traversal :
1 -> 4 -> 5 -> 6 -> 7 -> 3 ->

Levelorder Traversal :
3 -> 1 -> 7 -> 6 -> 5 -> 4 ->

Internal Node in BST : 4

External Node in BST : 2

Total Node in BST : 6

Height Node in BST : 4
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9> ▯
```

2. Write a program to do the following operations.

● Create an array from user input.

● Search Element in an array using linear search - prints iteration done to find the element

● Search Element in an array using binary search - prints iteration done to find the element

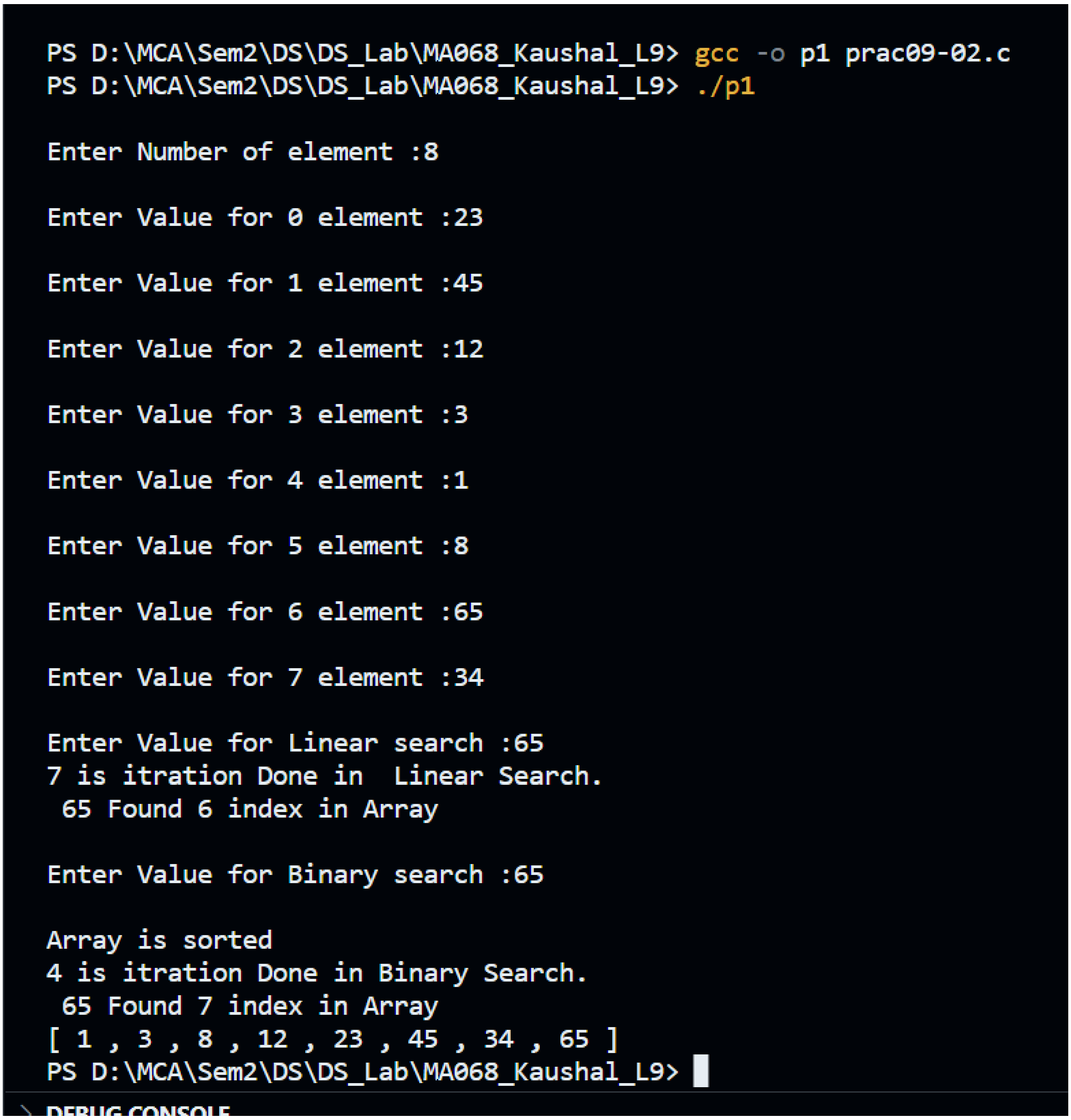
Code:

```
#include <stdio.h>


int linearSearch(int arr[],int n,int key){

   int i,count=0;
   for(i=0; i<n; i++)
   {
      if(key==arr[i])
      {
            printf("%d is itration Done in  Linear Search.",i+1);
            return i;
      }
   }
   printf("%d is itration Done in  Linear Search.",i);
   return -1;;
}



void bubbleSort(int arr[],int n){

   for(int i=0; i<n; i++)
      for(int j=0; j<i; j++)
         if(arr[j]>arr[j+1])
         {
            int temp=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=temp;
         }

}
```

```
int binarySearch(int arr[],int n,int key){

   int l=0;
   int r=n-1;
   int count=0;

   while (l <= r) {
      //int m = l + (r - l) / 2;
      int m=(r+l)/2;
      count++;
      if (arr[m] == key)
      {
         printf("%d is itration Done in Binary Search.",count);
         return m;
      }

      if (arr[m] < key)
         l = m + 1;
      else
         r = m - 1;
   }
   printf("%d is itration Done in Binary Search.",count);
   return -1;
}










int main(){

   int n;

   //Create Array
```

```
    printf("\nEnter Number of element :");
    scanf("%d",&n);

    int arr[n];
    for(int i=0; i<n; i++){
        printf("\nEnter Value for %d element :",i);
        scanf("%d",&arr[i]);
    }

    int Lkey;
    printf("\nEnter Value for Linear search :");
    scanf("%d",&Lkey);

    int FoundIndexL=linearSearch(arr,n,Lkey);
    if(FoundIndexL==-1)
        printf("\n %d not Found in Array",Lkey);
    else
        printf("\n %d Found %d index in Array",Lkey,FoundIndexL);


    int Bkey;

    printf("\n\nEnter Value for Binary search :");
    scanf("%d",&Bkey);

    bubbleSort(arr,n);
    printf("\nArray is sorted\n");

    int FoundIndexB=binarySearch(arr,n,Bkey);
    if(FoundIndexB==-1)
        printf("\n %d not Found in Array",Bkey);
    else
        printf("\n %d Found %d index in Array",Bkey,FoundIndexB);

    //Print array
    printf("\n[");
    for(int i=0; i<n; i++){
        printf(" %d ,",arr[i]);
    }
    printf("\b]");

    return 0;
}
```

Output :



```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9> gcc -o p1 prac09-02.c
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9> ./p1

Enter Number of element :8

Enter Value for 0 element :23

Enter Value for 1 element :45

Enter Value for 2 element :12

Enter Value for 3 element :3

Enter Value for 4 element :1

Enter Value for 5 element :8

Enter Value for 6 element :65

Enter Value for 7 element :34

Enter Value for Linear search :65
7 is itration Done in  Linear Search.
 65 Found 6 index in Array

Enter Value for Binary search :65

Array is sorted
4 is itration Done in Binary Search.
 65 Found 7 index in Array
[ 1 , 3 , 8 , 12 , 23 , 45 , 34 , 65 ]
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L9>
```