1.***Write a C program to print preorder, inorder, and postorder traversal on Binary Tree.******

```
#include <stdio.h>
#include <stdlib.h>
void Postorder();
void Inorder();
void Preorder();
struct node
{
    int data;
    struct node* left;
    struct node* right;
};
struct node* newNode(int data)
{
    struct node* node = (struct node*)
      malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

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

```
void main()
{
     struct node *root  = newNode(1);
     root->left             = newNode(2);
     root->right            = newNode(3);
     root->left->left     = newNode(4);
     root->left->right   = newNode(5);

    printf("\nPreorder traversal of binary tree is \n");
    Preorder(root);
    printf("\nInorder traversal of binary tree is \n");
    Inorder(root);
    printf("\nPostorder traversal of binary tree is \n");
    Postorder(root);

}
```

2.***Write a C program to create (or insert) and inorder traversal on Binary Search Tree*******

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int key;
    struct node *left, *right;
};
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
void inorder(struct node *root)
{
    if (root != NULL)
    {
    inorder(root->left);
    printf("%d \n", root->key);
    inorder(root->right);
    }
}
struct node* insert(struct node* node, int key)
```

```
{
   if (node == NULL) return newNode(key);
   if (key < node->key)
   node->left  = insert(node->left, key);
   else if (key > node->key)
   node->right = insert(node->right, key);
   return node;
}

int main()
{
   struct node *root = NULL;
   root = insert(root, 3);
   insert(root, 12);
   insert(root, 51);
   insert(root, 43);
   insert(root, 37);
   insert(root, 98);
   insert(root, 5);

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

3.***Write a C program to create (or insert) and inorder traversal on Binary Search Tree******

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int a[10],i,size,item,pos,flag=0;

   printf("\n  Enter the size of an array: ");
   scanf("%d",&size);

   printf("\n  Enter the elements of the array: ");

   //LOOP TO STORE THE ELEMENTS
   for(i=0;i<size;i++)
   {
      scanf("%d",&a[i]);
   }

   printf("\n  Enter the element to be searched: ");
```

```
    scanf("%d",&item);


    for(i=0;i<size;i++)
    {
        if(item==a[i])
        {
            pos=i;
            flag=1;
            break;
        }
    }

    if(flag==1)
        printf("\n  The element is in the list and its position is: %d",pos+1);
    else
        printf("\n  The element is not found");

}
```

4.***Write a C program for binary search algorithm.*******

```
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
    return -1;
}

void main(void)
{
    int arr[] = { 5,3,1,54,24,23,};
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? printf("Element is not present in array")
                   : printf("Element is present at index %d", result);
}
```