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

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

typedef struct Binarytree
{
   int data;
   struct Binarytree *l;
   struct Binarytree *r;
} nod;

nod *create();
void insert(nod *, nod *);
void preorder(nod *);
void postorder(nod *);
void inorder(nod *);

int main()
{
   int var;
   nod*root = NULL, *temp, *current;
   printf("Enter the number of Nodes you want to be in binarytree :");
   scanf("%d", &var);
   printf("Enter %d Nodes data ",var);
   do
   {
      temp = create();
      if (root == NULL)
         root = temp;
      else
         insert(root, temp);
      var--;

   } while (var != 0);

   printf("Preorder");
   preorder(root);

   printf("Inorder");
   inorder(root);

   printf("Postorder");
   postorder(root);
```

```
    return 0;
}

nod *create()
{
    nod *ano;

    ano= (nod *)malloc(sizeof(nod));
    scanf("%d", &ano->data);
    ano->l = ano->r = NULL;
    return ano;
}

void insert(nod *root, nod *tre)
{
    if (root == NULL)
    {
        root = tre;
    }
    else
    {

        if (tre->data < root->data)
        {
            if (root->l != NULL)
                insert(root->l, tre);
            else
                root->l = tre;
        }

        if (tre->data > root->data)
        {
            if (root->r != NULL)
                insert(root->r, tre);
            else
                root->r = tre;
        }
    }
}

void preorder(nod *root)
```

```
{
   if (root != NULL)
   {
      printf("%d  ", root->data);
      preorder(root->l);
      preorder(root->r);
   }
}
```

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, 4);
   insert(root, 15);
   insert(root, 51);
   insert(root, 43);
   insert(root, 32);
   insert(root, 29);
   insert(root, 7);

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

3.*****Write a C program for linear search algorithm.********

```
#include <stdio.h>

int main()
{

   int number,x,k, val_find, found = 0;
   printf("Enter the number of elements that u want to be in the array: ");
   scanf("%d", &number);
   int arr[number];
   printf("Enter the elements sequentially: \n");
   for (k = 0; k< number; k++)
   {
      scanf("%d", &arr[k]);
   }

   printf("Enter the element to be searched: ");
   scanf("%d", &val_find);

   for (x = 0; x< number; x++)
   {
      if (val_find == arr[x])
      {
         found = 1;
         break;
      }
   }
```

```
    if (found == 1)
        printf("Element is there in the array at the position %d", x );
    else
        printf("Element isn't  there in the array\n");

    return 0;
}
```

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);
}
```