PROGRAMS

/*1)C program to print preorder, in order, and postorder traversal on Binary Tree*/

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

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);
}
 int main()
{
    struct node *root  = newNode(5);
    root->left          = newNode(9);
    root->right         = newNode(1);
    root->left->left    = newNode(7);
    root->left->right   = newNode(4);

    printf("\nThe Preorder,Inorder,Post order for the given number are as follows:");
    printf("\n\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);

    getchar();
    return 0;
}
```

OUTPUT

The Preorder,Inorder,Post order for the given number are as follows:

```
Preorder traversal of binary tree is
5 9 7 4 1
Inorder traversal of binary tree is
7 9 4 5 1
Postorder traversal of binary tree is
7 4 9 1 5
```

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

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

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 Inorder(struct node* node)
{
    if (node == NULL)
        return;

    Inorder(node->left);

    printf("%d ", node->data);

    Inorder(node->right);
}
 int main()
{
```

```
    struct node *root  = newNode(5);
    root->left            = newNode(9);
    root->right           = newNode(1);
    root->left->left     = newNode(7);
    root->left->right   = newNode(4);

    printf("\nInorder traversal of binary tree is \n");
    Inorder(root);
}
```

OUTPUT

```
Inorder traversal of binary tree is
7 9 4 5 1
```

```
/*3)Write a C program for linear search algorithm.*/
#include<stdio.h>

#include<conio.h>

void main(){
  int list[20],size,i,sElement;

  printf("Enter size of the list: ");
  scanf("%d",&size);

  printf("Enter any %d integer values: ",size);
  for(i = 0; i < size; i++)
    scanf("%d",&list[i]);

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

  // Linear Search Logic
  for(i = 0; i < size; i++)
  {
     if(sElement == list[i])
     {
        printf("Element is found at %d index", i);
        break;
     }
  }
  if(i == size)
```

```
    printf("Given element is not found in the list!!!");
  getch();
}
```

OUTPUT

```
Enter size of the list: 5
Enter any 5 integer values: 3 5 7 3   9 1
Enter the element to be Search: 7
Element is found at 2 index
```

/*4)Write a C program for binary search algorithm*/

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

void main()
{
   int first, last, middle, size, i, sElement, list[100];


   printf("Enter the size of the list: ");
   scanf("%d",&size);

   printf("Enter %d integer values in Assending order\n", size);

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

   printf("Enter value to be search: ");
   scanf("%d", &sElement);

   first = 0;
   last = size - 1;
   middle = (first+last)/2;

   while (first <= last) {
      if (list[middle] < sElement)
         first = middle + 1;
      else if (list[middle] == sElement) {
         printf("Element found at index %d.\n",middle);
         break;
      }
      else
         last = middle - 1;
```

```
      middle = (first + last)/2;
    }
    if (first > last)
      printf("Element Not found in the list.");
    getch();
}
```

OUTPUT

```
Enter the size of the list: 4
Enter 4 integer values in Assending order
1 2 3 4
Enter value to be search: 3
Element found at index 2.
```