/*1.Write a C program to print preorder, inorder, 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 printPostorder(struct node* node)
{
    if(node==NULL)
    return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d",node->data);
}
void printInorder(struct node* node)
{
   if(node==NULL)
    return;
   printInorder(node->left);
   printf("%d",node->data);
   printInorder(node->right);
}
void printPreorder(struct node* node)
{
    if(node==NULL)
    return;
    printf("%d",node->data);
```

```
    printPreorder(node->left);
    printPreorder(node->right);
}
int 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");
    printPreorder(root);
    printf("\nInorder traversal of binary tree is\n");
    printInorder(root);
    printf("\nPostorder traversal of binary tree is\n");
    printPostorder(root);

    getchar();
    return 0;
}
```

OUTPUT:

```
Preorder traversal of binary tree is
12453
Inorder traversal of binary tree is
42513
Postorder traversal of binary tree is
45231
```

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

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

typedef struct node {
  int data;
  struct node *left;
  struct node *right;
  struct node *parent;
}node;
```

```
typedef struct binary_search_tree {
  node *root;
}binary_search_tree;

node* new_node(int data) {
  node *n = malloc(sizeof(node));
  n->data = data;
  n->left = NULL;
  n->right = NULL;
  n->parent = NULL;

  return n;
}

binary_search_tree* new_binary_search_tree() {
  binary_search_tree *t = malloc(sizeof(binary_search_tree));
  t->root = NULL;

  return t;
}

node* minimum(binary_search_tree *t, node *x) {
  while(x->left != NULL)
    x = x->left;
  return x;
}

void insert(binary_search_tree *t, node *n) {
  node *y = NULL;
  node *temp = t->root;
  while(temp != NULL) {
    y = temp;
    if(n->data < temp->data)
      temp = temp->left;
    else
      temp = temp->right;
  }
  n->parent = y;

  if(y == NULL) //newly added node is root
    t->root = n;
  else if(n->data < y->data)
    y->left = n;
```

```
  else
    y->right = n;
}

void inorder(binary_search_tree *t, node *n) {
  if(n != NULL) {
    inorder(t, n->left);
    printf("%d\n", n->data);
    inorder(t, n->right);
  }
}

int main() {
  binary_search_tree *t = new_binary_search_tree();

  node *a, *b, *c, *d, *e, *f, *g, *h, *i, *j, *k, *l, *m;

  a = new_node(10);
  b = new_node(20);
  c = new_node(30);
  d = new_node(100);
  e = new_node(90);
  f = new_node(40);
  g = new_node(50);
  h = new_node(60);
  i = new_node(70);
  j = new_node(80);
  k = new_node(150);
  l = new_node(110);
  m = new_node(120);

  insert(t, a);
  insert(t, b);
  insert(t, c);
  insert(t, d);
  insert(t, e);
  insert(t, f);
  insert(t, g);
  insert(t, h);
  insert(t, i);
  insert(t, j);
  insert(t, k);
  insert(t, l);
```

```
  insert(t, m);



  inorder(t, t->root);
  return 0;
}
```

OUTPUT:
10
20
30
40
50
60
70
80
90
100
110
120
150

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

```
#include<stdio.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: ");


    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");
    getch();
}
```

OUTPUT:

```
  Enter the size of an array: 6

  Enter the elements of the array: 1 2 3 4 5 6

  Enter the element to be searched: 4

  The element is in the list and its position is: 4
```

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

```
#include<stdio.h>

void main()
{
```

```
    int a[10],i,n,item,flag=0,small,big,mid;

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

    printf("\n  Enter the elements in ascending order: ");
    for(i=0;i<n;i++)
    {
       scanf("%d",&a[i]);
    }

    printf("\n  Enter the number to be search: ");
    scanf("%d",&item);

    small=0,big=n-1;
    while(small<=big)
    {
       mid=(small+big)/2;
       if(item==a[mid])
       {
          flag=1;
          break;
       }
       else if(item<a[mid])
       {
          big=mid-1;
       }
       else
       small=mid+1;
    }
    if(flag==0)
    printf("\n  The number is not found");
    else
    printf("\n  The number is found and its position is: %d",mid+1);
    getch();
}
```

OUTPUT:

```
  Enter the size of an array: 5

  Enter the elements in ascending order: 11 12 13 14 15
```

Enter the number to be search: 4

The number is not found