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 *
createNode (int value)
{
struct node *newNode = malloc (sizeof (struct node));
newNode->data = value;
newNode->left = NULL;
newNode->right = NULL;

return newNode;
}
struct node *
insert (struct node *root, int data)
{
if (root == NULL)
return createNode (data);

if (data < root->data)
root->left = insert (root->left, data);
else if (data > root->data)
root->right = insert (root->right, data);

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);
}
int findmin(struct node* root)
{
if(root==NULL)
{

    return -1;

}
else if(root->left==NULL)
{
    return root->data;
}

   return findmin(root->left);
}
int findmax(struct node* root)
{
if(root==NULL)
{

    return -1;
}
else if(root->right==NULL)
{
    return root->data;
}

   return findmax(root->right);
}
int findheight(struct node* root)
{
int x,y;
if(root==NULL)
{
return-1;
}
x=findheight(root->left);
y=findheight(root->right);
if(x>y)
return x+1;
else
return y+1;
}

int
main ()
{
struct node *root = NULL;
root = insert (root, 10);
root=insert(root,70);
root=insert(root,60);
root = insert (root, 30);
root = insert (root, 11);
root = insert (root, 60);

root = insert (root, 12);
root = insert (root, 14);
root = insert (root, 4);
root = insert (root, 50);
root = insert (root, 358);
root = insert (root, 40);
printf("inorder traversal");
inorder(root);
printf("\npreorder traversal");
```

```
preorder(root);
printf("\npostorder traversal");
postorder(root);


}
```

Output:

```
inorder traversal:4 -> :10 -> :11 -> :12 -> :14 -> :30 -> :40 -> :50 -> :60 -> :70 -> :358 ->
preorder traversal :10 -> :4 -> :70 -> :60 -> :30 -> :11 -> :12 -> :14 -> :50 -> :40 -> :358 ->
postorder traversal:4 ->:14 ->:12 ->:11 ->:40 ->:50 ->:30 ->:60 ->:358 ->:70 ->:10 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

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 data;
   struct node* left;
   struct node* right;
};

struct node* createNode(value){
   struct node* newNode = malloc(sizeof(struct node));
   newNode->data = value;
   newNode->left = NULL;
   newNode->right = NULL;

   return newNode;
}


struct node* insert(struct node* root, int data)
{
   if (root == NULL) return createNode(data);

   if (data < root->data)
```

```
        root->left  = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);

    return root;
}

void inorder(struct node* root){
    if(root == NULL) return;
    inorder(root->left);
    printf("%d ->", root->data);
    inorder(root->right);
}


int main(){
    struct node *root = NULL;
    root = insert(root, 8);
    insert(root, 3);
    insert(root, 1);
    insert(root, 6);
    insert(root, 7);
    insert(root, 10);
    insert(root, 14);
    insert(root, 4);

    inorder(root);
}
```

Output:

```
1 ->3 ->4 ->6 ->7 ->8 ->10 ->14 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

3)Write a C program for linear search algorithm.

```
#include<stdio.h>
int main()
{
    int a[5],i,n,se,found=0;
    printf("enter the number of variables to be used\n");
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        printf("enter the value of a[%d]\n",i);
        scanf("%d",&a[i]);
    }
    printf("enter the searching element\n");
    scanf("%d",&se);
    for (i=0;i<n;i++)
    {
        if (a[i]==se)
        {
            printf("element found at %d position\n",i);
            break;
        }
    }
    if (i==n);
    {
        printf("element not found\n");
    }
    return 0;
}
```

Output:

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

    Language
main.c

```
input
6
enter the number of variables to be used
5
enter the value of a[0]
23
enter the value of a[1]
```

```
75
enter the value of a[2]
85
enter the value of a[3]
  96
enter the value of a[4]
100
enter the searching element
96
element found at 3 position
```

4)Write a C program for binary search algorithm

```
#include<stdio.h>
int main()
{
    int a[10],n,i,se,found=0,top,mid,bot;
    printf("enter the number of variables\n");
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        printf("enter the value of a[%d]\n",i);
        scanf("%d",&a[i]);
    }
    printf("enter the searching element\n");
    scanf("%d",&se);
    top=0;
    bot=n-1;
    while(top<=bot)
    {
        mid=(top/bot)/2;
        if (a[mid]==se)
        {
            found=1;
            break;
        }
        else if (a[mid>se])
        {
            bot=mid-1;
        }
        else if (a[mid<se])
        {
            top=mid+1;
        }
    }
    if (found==1)
    {
        printf("element found at %d position\n",mid);
    }
    else
    {
        printf("element not found");
    }
    return 0;
}
```

Output:

```
enter the number of variables
5
enter the value of a[0]
23
enter the value of a[1]
46
enter the value of a[2]
12
34
enter the value of a[4]
27
enter the searching element
23
element found at 0 position
```