ASSIGNMENT

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;
        int value;
        struct node* left;
        struct node* right;
};

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);
}
```

ASSIGNMENT

```
void postorder(struct node* root) {
        if(root == NULL) return;
    postorder(root->left);
    postorder(root->right);
        printf("%d ->", root->data);
}
struct node *createNode(value)
{
        struct node* newNode = malloc(sizeof(struct node));
        newNode->data = value;
        newNode->left = NULL;
        newNode->right = NULL;

        return newNode;
}
void main()
{
        struct node* root = createNode(1);
    root->left=createNode(12);
    root->right=createNode(9);

    root->left->left=createNode(10);
    root->left->right=createNode(15);

    root->right->left=createNode(11);
```

ASSIGNMENT

```
    root->right->right=createNode(16);



    printf("Inorder traversal \n");
        inorder(root);


    printf("\nPreorder traversal \n");
        preorder(root);


    printf("\nPostorder traversal \n");
        postorder(root);
}
```

Output:

```
Inorder traversal
10 ->12 ->15 ->1 ->11 ->9 ->16 ->
Preorder traversal
1 ->12 ->10 ->15 ->9 ->11 ->16 ->
Postorder traversal
10 ->15 ->12 ->11 ->16 ->9 ->1 ->
```

2)write a c program to create (or insert) and inorder traversal on binary search tree

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

ASSIGNMENT

```
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)
```

ASSIGNMENT

```
        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;
}
```

Output:

```
37
43
51
98
```

ASSIGNMENT

3) write c program for linear search algorithm

```
#include <stdio.h>
int main()
{
        int array[100],search,c,n;
        printf("enter number of elements in array\n");
   scanf("%d",&n);
        printf("enter %d integer(s)\n",n);
   for(c=0;c<n;c++)
    scanf("%d",&array[c]);
 printf("enter a number to search\n");
 scanf("%d",&search);
 for(c=0;c<n;c++)
 {
    if(array[c]==search)
        {
       printf("%d present at ;location %d\n",search,c+1);
        break;
        }
 }
 if(c==n)
  printf("%d isn't present in the array\n",search);
```

```
 return 0;
}
```

Output:

```
enter number of elements in array
5
enter 5 integer(s)
25
14
36
95
38
enter a number to search
95
95 present at ;location 4
```

4)write a c program for binary search algorithm

```
#include <stdio.h>
int main()
{
 int c,first,last,middle,n,search,array[100];
 printf("enter number of elements\n");
 scanf("%d",&n);
 printf("enter %d integers\n",n);
 for(c=0;c<n;c++)
```

ASSIGNMENT

```
  scanf("%d",&array[c]);
 printf("enter value to find/n");
 scanf("%d",&search);
 first=0;
 last=n-1;
 middle=(first+last)/2;
 while(first<=last){
   if(array[middle]<search)
          first=middle+1;
          else if(array[middle]==search){
       printf("%d found at location %d\n",search,middle+1);
          break;
          }
          else
          last=middle-1;
       middle=(first+last)/2;
 }
 if(first>last)
          printf("not found! %d isn;t present in the list\n",search);
 return 0;
}
```

Output:

```
enter number of elements
5
```

ASSIGNMENT

```
enter 5 integers

25

36

41

51

95

enter value to find/n51

51 found at location 4
```