Program 1:

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

```
#include<stdlib.h>
struct node{
int data;
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);
}
```

```
void postorder(struct node* root){
if (root==NULL)return;
postorder(root->left);
postorder(root->right);
printf("%d->",root->data);
}
struct node* createNode(int 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->right->right=createNode(15);
printf("inorder traversal \t");
inorder(root);

printf("\npreorder traversal \t");
preorder(root);

printf("\npostorder traversal \t");
postorder(root);
}
```

Output 1:

```
inorder traversal    10->12->1->9->15->
preorder traversal 1->12->10->9->15->
postorder traversal      10->12->15->9->1->
```

Program 2:

Write a C program to create (or insert) and in order 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 item) {
struct node*temp = (struct node*)malloc(sizeof(struct node));
temp->data = item;
temp->left = temp->right = NULL;
return temp;
}
struct node* insert(struct node *node, int value){
if (node==NULL)return newNode(value);
if (value<node->data)
node->left = insert(node->left, value);
else if(value>node->data)
node->right = insert(node->right, value);
return node;
}
void inorder (struct node* root) {
if(root == NULL) return;
inorder(root->left);
printf("%d->", root->data);
inorder(root->right);
}
void main () {
struct node* root = NULL;
root = insert(root, 50);
insert(root, 30);
insert(root, 20);
insert(root, 40);
insert(root, 70);
insert(root, 80);
insert(root, 60);
printf("\n inorder traversal \n");
inorder(root);
}
```

Output 2:

```
 In order traversal
20->30->40->50->60->70->80->
```

Program 3:

Write a C program depth first search (DFS) using array.

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v) {
	int i;
	reach[v]=1;
	for (i=1;i<=n;i++)
	  if(a[v][i] && !reach[i]) {
		printf("\n %d->%d",v,i);
		dfs(i);
	}
}
```

```
void main() {
    int i,j,count=0;
    printf("\n Enter number of vertices:");
    scanf("%d",&n);
    for (i=1;i<=n;i++) {
        reach[i]=0;
        for (j=1;j<=n;j++)
          a[i][j]=0;
    }
    printf("\n Enter the adjacency matrix:\n");
    for (i=1;i<=n;i++)
     for (j=1;j<=n;j++)
      scanf("%d",&a[i][j]);
    dfs(1);
    printf("\n");
    for (i=1;i<=n;i++) {
        if(reach[i])
          count++;
    }
    if(count==n)
     printf("\n matrix is connected");
    else
      printf("\n matrix is not connected");
    getch();
}
```

Output 3:

```
Enter the number of vertices: 2
Enter the adjacency matrix:
2
5
8
6
1->2
Matrix is connected
```

Program 3 (Linear search):

Write a C program for linear search algorithm.

```
#include<stdio.h>
main()
{
	int a[20],i,n,s,flag=0;
	printf("enter the no elements of array");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("enter %d element of array :",i+1);
		scanf("%d",&a[i]);
	}
	printf("enter the element to search:");
	scanf("%d",&s);
	for(i=0;i<n;i++)
	{
		if(a[i]==s)
		{
			printf("element found");
			flag=1;
```

```
                }
        }
        if(flag==0)
                printf("element not found");
}
```

Output 3:

```
Enter the no elements of array
6
Enter sorted array only:
Enter 1 element of array: 10
Enter 2 element of array: 20
Enter 3 element of array: 30
Enter 4 element of array: 40
Enter 5 element of array: 50
Enter 6 element of array: 60
Enter the element to search: 20
Element found
```

Program4 (binary search):

Write a C program for binary search algorithm

```
#include<stdio.h>
main()
{
	int a[20],first,n,s,middle,last;
	printf("enter the no elements of array\n");
	scanf("%d",&n);
	printf("enter sorted array only:\n");
	for(first=0;first<n;first++)
	{
		printf("enter %d element of array :",first+1);
		scanf("%d",&a[first]);
	}
	printf("enter the element to search:");
	scanf("%d",&s);
	first=0;
	last=n-1;
	while(first<=last)
	{
		middle=(first+last)/2;
```

```
        if(a[middle]==s)
        {
            printf("element found");
            break;
        }
        else
        {
            if(s<a[middle])
            {
                last=middle-1;
            }
            else
            {
                first=middle+1;
            }
        }
    }
    if(first>last)
    {
        printf("element not found");
    }
```

```
}
```

Output 4:

```
Enter the no elements of array
8
Enter sorted array only:
Enter 1 element of array: 5
Enter 2 element of array: 8
Enter 3 element of array: 17
Enter 4 element of array: 24
Enter 5 element of array: 36
Enter 6 element of array: 57
Enter 7 element of array: 61
Enter 8 element of array: 78
Enter the element to search: 8
Element found
```