// 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(value) {

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

 return newNode;
 }

Structnode* 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,40);
     insert (root,100);
      insert (root,300);
       insert (root,600);
        insert (root,500);
        insert (root, 900);
        insert (root,1000);
       insert (root,400);
     inorder(root);
  }
```

// C program to create (or insert) and inorder traversal on Binary Search tree

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

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

   return new Node;
  }
   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, 18);
    insert (root, 113);
    insert (root, 111);
    insert (root, 161);
    insert (root, 117);
    insert (root, 190);
    insert (root, 150);
    insert (root, 141);

    inorder (root);
}
```

//write a C program depth first search (DFS) using array

```
#include <stdio.h>
#include <stdlib.h>
int source, V, E, time, visited [100], G[100][100];
void DFS(int i)
 {
   int j;
   visited[i]=1;
   Printf("%d->", i+1);
   for (j=0; j<V, j++)
   {
      if (G[i][j] ==1 && visited[j]==0)
        DFS(j);
      }
   }
  intmain()
    {
     int i,j,V1,V2;
    Printf("\t\t\t Graphs\n");
   Printf("Enter the no. of Edges:");
   Scanf("%d", &E);
    Printf("Enter no. of vertices :");
    Scanf("%d", &V);
    for (i=0; i<V; i++)
    {
      for (j=0; j<V; j++)
          G[i][j]=0;
    }
```

```
/* Creating edges: */
for (i=0; i<E; i++)
{
 Printf("Enter the edges (format: V1V2):");
 Scanf("%d%d", &V1, &V2);
  G[V1-1][V2-1]=1;
 }
 for (i=0; i<V; i++)
 {
   for (j=0; j<V; j++)
      Printf("%d", G[i][j];
   Print("\n");
 }
 Print("Enter the source:");
 Scanf("%d", &source);
      DFS(source-1);
   return 0;
}
```

Write a C program breath first search (BFS) using array ④

```
#include <stdio.h>
int G[100][100], q[100], visited[100], n, front = 1, rear = 0;
void bfs (int v)
{
  int i;
   visited [v] = 1;
for (i = 1; i <= n; i++)
if (G[v][i] && !visited [i])
 q [++rear] = i;
  if (front <= rear)
  bfs (q[front++]);
 }
int main ()
{
int v; i, j;
Printf ("\n Enter the number of vertices:");
 scanf ("%d", &n);
 for (i = 1; i <= n; i++)
{
  q[i] = 0;
   visited[i] = 0;
  }
   Printf ("\n Enter graph data in matrix form:\n");
   for (i = 1; i <= n; i++)
   for (j = 1; j <= n, j++)
    scanf ("%d", &G[i][j]);
   Printf ("\n Enter the starting vertex:");
   Scanf ("%d", &v);
```

```
bfs(v);
Printf("\n The nodes which are reachables are:\n");
  for (i=1;i<=n;i++)
   if (visited [i])
   Printf ("%d\t",i);
   else
     Printf ("\n %d is not reachable",i);
 return 0;
}
```

// C program for linear search algorithm

```
#include <stdio.h>
int main()
{
    int a[100], i, x, n;
    Printf ("How many elements?");
     Scanf ("%d", &n);

    Printf ("Enter array elements: \n");
     for (i=0; i<n; ++i)
          scanf ("%d", &a[i]);
     Printf ("\n Enter element to search:");
      scanf ("%d", &x);
     for (i=0; i<n; ++i)
        if (a[i]==x)
            break;
      if (i<n)
          Printf("Element found at index %d", i);
      else
          Printf ("Element not found");
      return 0;
}
```

11c Program for binary search algorithm

```
#include <stdio.h>
int main()
{
int arr[100], i, n, x, flag=0, first, last, mid;

Printf("enter size of array:");
Scanf("%d",&n);
Printf("\n enter array element (ascending order) \n");

for (i=0;i<n;++i)
  Scanf("%d", &arr[i]);

Printf("\nEnter the element to search:");
Scanf("%d", &x);
first = 0;
last = n-1;
while (first<=last)
{
  mid = (first + last)/2;
  if (x==arr[mid]) {
      flag = 1;
      break;
  }
  else
     if (x>arr[mid])
        first = mid+1;
     else
        last = mid-1;
}
```

```
if (flag == 1)
    Printf ("\n Element found at position %d", mid+1);
else
    Printf ("\n Element not found");
return 0;
}
```