1.

```
#include < stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* left;
    struct node* right;
};
struct node* new Node (int data)
{
    struct node* node = (struct node*)
                        malloc (size of (struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}
void print Post order (struct node* node)
{
    if (node == NULL)
      return;
    print Post order(node->left);
    print Post order(node->right);
    print f ("%d ", node->data);
}
void print In order (struct node* node)
```

```
{
    if (node == NULL)
        return;
    print In order(node->left);
    print f ("%d ", node->data);
    print In order(node->right);
}
void print Pre order (struct node* node)
{
    if (node == NULL)
        return;
    print f ("%d ", node->data);
    print Pre order(node->left);
    print Pre order(node->right);
}
int main ()
{
    struct node * root   = new Node (0);
    root->left           = new Node (2);
    root->right          = new Node (5);
    root->left->left     = new Node (7);
    root->left->right    = new Node (9);

    print f("\n Pre order traversal of binary tree is \n");
    print Pre order (root);

     print f ("\n In order traversal of binary tree is \n");
    print In order(root);
```

```
    print f("\n Post order traversal of binary tree is \n");
    print Post order(root);
    return 0;
}
```

Output :-

Pre order traversal of binary tree is

0 2 7 9 5

In order traversal of binary tree is

7 2 9 0 5

Post order traversal of binary tree is

7 9 2 5 0

….. program finished with exit code 0

Press ENTER to exit console.

2.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int key;
   struct node *left, *right;
};
struct node *new Node (int item)
{
   struct node *temp = (struct node *)malloc (size of(struct node));
   temp->key = item;
   temp->left = temp->right = NULL;
```

```
    return temp;
}
void in order (struct node *root)
{
    if (root! = NULL)
    {
        In order(root->left);
        Print f("%d \n", root->key);
        In order(root->right);
    }
}
struct node* insert (struct node* node, int key)
{
  if (node == NULL) return new Node(key);
  if (key < node->key)
        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, 10);
    insert (root, 20);
    insert (root, 30);
    insert (root, 50);
```

```
   insert (root, 70);
   insert (root, 80);
   insert (root, 100);
   in order(root);
   return 0;
}
```

Output :-

```
10
20
30
50
70
80
100
… Program finished with exit code 0
Press ENTER to exit console.
```

3.

```
#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]) {
                print f("\n %d->%d",v,i);
                dfs(i);
```

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

```
Enter number of vertices : 2

Enter the  adjacency matrix :

1

1

1

0

1->2

Matrix is connected

…. Program finished with exit code 255

Press ENTER to exit console.
```

4.

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v) {
        for (i=1;i<=n;i++)
          if(a[v][i] && !visited[i])
            q[++r]=i;
        if(f<=r) {
                visited[q[f]]=1;
                bfs(q[f++]);
        }
}
void main() {
        int v;
        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",&a[i][j]);
        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
        bfs(v);
        printf("\n The node which are reachable are:\n");
        for (i=1;i<=n;i++)
          if(visited[i])
            printf("%d\t",i); else
            printf("\n Bfs is not possible");
        getch();
}
```

Output:-

```
Enter the number of vertices : 3
Enter graph data is matrix form :
1
0
0
0
1
```

0

0

0

1

Enter the starting vertex:2

The node which are reasonable are:

Nfs is not possible2

Nfs is not possible

… program finished with exit code 255

Press ENTER to exit console.

5.

```
#include <stdio.h>
int main()
{
  int array[100], search, i, n;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integers\n",n);

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

  printf("Enter a number to search\n");
  scanf("%d", &search);
```

```
 for (i = 0; i < n; i++)
 {
   if (array[i] == search)
   {
     printf("%d is found in the array %d.\n", i+1);
     break;
   }
 }
 if (i == n)
   printf("%d is not found in the array.\n");

 return 0;
}
```

Output:-

```
Main.c:21:41: warning : format ‘%d’ expects a matching ‘int’ argument [-Wformat=]
Main.c:26:14: warning : format ‘%d’ expects a matching ‘int’ argument [-Wformat=]
Inter number of elements in array
4
Enter 4 integers 1
2
3
4
Enter a number to search
2
2 is found in the array 2.
… program finished with exit code 0
```

Press ENTER to exit console.

6.

```
#include <stdio.h>
int main()
{
 int i,first,last,middle,n,search,array[100];

 printf("Enter the number of elements\n");
 scanf("%d", &n);

 printf("Enter %d elements\n", n);

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

 printf("Enter the element to search\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 is found in the array %d.\n", middle+1);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if (first > last)
    printf("%d is not found in the array.\n", search);

  return 0;
}
```

Output:-

```
Main.c:25:41:warning: format ‘%d’ expects a matching ‘int’ argument [-wformat=]
Enter the number of elements
4
Enter 4 elements
1
2
3
4
Enter the element to search
4
4 is found in the array 4.

… program finished with code 0
```

Press ENTER to exit console.