```c
/*
DS11:Design, Develop and Implement a Program in C for the following operations on
     Graph(G) of Cities
a.Create a Graph of N cities using Adjacency Matrix.
b.Print all the nodes reachable from a given starting node in a digraph using
DFS/BFS method.

The program contains 3 functions:
DFS() function
BFS() function
main() function

DFS:
Input : Adjacency matrix representation of the graph.
output: Nodes/vertices connected
        Whether graph is connected or not.
BFS:
Input : Adjacency matrix representation of the graph.
        Starting vertex
output: All the nodes/vertices that can be reached from starting vertex.
*/

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

int a[10][10];          // Two dimensional array for adjacency matrix
int q[10];             // Queue used for BFS function.
int visited[10];      // Stores all visited nodes.
int reach[10];       // Stores final reached nodes
int n;              // Number of nodes
int i,j;
int f=0,r=-1;    // f:front,r:rear (Used in queue function of BFS)

int count=0;   //Stores the number of nodes visited.
/*
if count == n-1 then all the nodes in a graph is connected.
otherwise the graph has node(s) that are not connected by any nodes.

if(0) means the statement following if condition will not be executed.
if(1) means the statement following if condition will be executed.
*/

void DFS(int v)                                 //DFS function
{
    int i; reach[v]=1;
    for(i=1;i<=n;i++)
        {
            if(a[v][i] && !reach[i])
            {
                printf("\n %d->%d",v,i);
                count++;
                DFS(i);                     //Recursive function call
            }
        }
}

void BFS(int v)                             //BFS function definition
{
    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++]);          //Recursive function call
```

```
                }
}

/*
For both DFS and BFS, the common input is -
number of vertices and adjacenecy matrix representing a graph.
*/
void main()
{
    int v, ch;

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

/*
i=1 means starting from 1st vertex.
Initially all values of 'q' array, 'visited' array and 'reach' array
are assigned with 0 value.
This value will change as we evaluate step by step.
*/
    for(i=1;i<=n;i++)
        {
            q[i]=0;
            visited[i]=0;
        }

    for(i=1;i<=n-1;i++)
        reach[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]);           //adjacency matrix

    printf("1.DFS\n 2.BFS\n 3.Exit\n");
    printf("Enter the choice\n");
    scanf("%d",&ch);


    switch(ch)
    {
        case 1: DFS(1);                    //Start from node 1
                if(count == n-1)
                    printf("\n Graph is connected");
                else
                    printf("\n Graph is not connected");
                    break;

        case 2: printf("\n Enter the starting vertex:");
                scanf("%d",&v);

                BFS(v);  //function call for BFS function with v value.
/*
if starting vertex 'v' is less than 1 or
greater than number of vertices 'n', then BFS is not possible.
*/
                if((v<1)||(v>n))
                {
                    printf("\n BFS not possible");
                }

                else
                    {
                        printf("\n The nodes which are reachable from %d are:\n",v);
                        for(i=1;i<=n;i++)
                            if(visited[i])
```

```
133                                 printf("%d\t",i);//Printing reachable nodes.
134                         }
135                         break;
136
137             case 3: exit(0);
138         }
139     }
```

**Output 1:**

```
Enter the number of vertices:4

Enter graph data in matrix form:
0 1 0 1
0 0 1 0
1 0 0 0
0 0 0 0

1.DFS
2.BFS
3.Exit
Enter the choice
1

1->2
2->3
1->4
Graph is connected


Enter the number of vertices:4

Enter graph data in matrix form:
0 1 0 0
0 0 1 0
1 0 0 0
0 0 0 0

1.DFS
2.BFS
3.Exit
Enter the choice
1

1->2
2->3
Graph is not connected


Enter the number of vertices:4

Enter graph data in matrix form:
0 1 0 1
0 0 1 0
1 0 0 0
0 0 0 0

1.DFS
```

```
2.BFS
3.Exit
Enter the choice
2

Enter the starting vertex:2
The nodes which are reachable from 1 are:
1       2       3       4
```

**Output 2:**

```
Enter the number of vertices:4

Enter graph data in matrix form:
0 0 1 0
0 0 1 1
1 1 0 1
0 1 1 0

1.DFS
2.BFS
3.Exit
Enter the choice
2

Enter the starting vertex:1
The nodes which are reachable from 1 are:
1       2       3       4


Enter the number of vertices:4

Enter graph data in matrix form:
0 0 1 0
0 0 1 1
1 1 0 1
0 1 1 0
1.DFS
2.BFS
3.Exit
Enter the choice
1

1->3
3->2
2->4
Graph is connected
```