

BFS Program:

```
#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX_VERTICES 50

typedef struct Graph_t
{
    int V;

    bool adj[MAX_VERTICES][MAX_VERTICES];
} Graph;

Graph* Graph_create(int V)
{
    Graph* g = malloc(sizeof(Graph));

    g->V = V;

    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            g->adj[i][j] = false;
        }
    }

    return g;
}

void Graph_destroy(Graph* g)
{
    free(g);
}

void Graph_addEdge(Graph* g, int v, int w)
```

```
{  
    g->adj[v][w] = true;  
}
```

```
void Graph_BFS(Graph* g, int s)
```

```
{  
    bool visited[MAX_VERTICES];  
    for (int i = 0; i < g->V; i++)  
    {  
        visited[i] = false;  
    }
```

```
    int queue[MAX_VERTICES];
```

```
    int front = 0, rear = 0;
```

```
    visited[s] = true;
```

```
    queue[rear++] = s;
```

```
    while (front != rear)
```

```
    {
```

```
        s = queue[front++];
```

```
        printf("%d ", s);
```

```
        for (int adjacent = 0; adjacent < g->V;
```

```
            adjacent++)
```

```
        {
```

```
            if (g->adj[s][adjacent] && !visited[adjacent])
```

```
            {
```

```
                visited[adjacent] = true;
```

```

        queue[rear++] = adjacent;
    }
}
}

int main()
{

    Graph* g = Graph_create(4);
    Graph_addEdge(g, 0, 1);
    Graph_addEdge(g, 0, 2);
    Graph_addEdge(g, 1, 2);
    Graph_addEdge(g, 2, 0);
    Graph_addEdge(g, 2, 3);
    Graph_addEdge(g, 3, 3);

    printf("Following is Breadth First Traversal (starting from vertex 2) \n");
    Graph_BFS(g, 2);
    Graph_destroy(g);

    return 0;
}

```

Output:

```

Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1

```

DFS Program:

```
#include<stdio.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);
        }
}

int 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 Graph is connected");  
else  
    printf("\n Graph is not connected");  
return 0;  
}
```

Output

```
Enter number of vertices:8  
  
Enter the adjacency matrix:  
0 1 0 0 0 0 0 1 0  
1 0 1 0 0 0 0 1 0  
0 1 0 1 0 1 0 0 1  
0 0 0 1 0 1 0 0 0  
0 0 1 0 1 0 1 0 0  
0 0 0 1 0 1 0 1 1  
1 1 0 0 0 0 1 0 1  
0 0 1 0 0 0 1 1 0  
  
1->2  
2->4  
4->3  
3->6  
3->8  
8->5  
5->7  
  
Graph is connected
```