

```

#include <stdio.h>
#define MAX 10

int graph[MAX][MAX], visited[MAX];
int queue[MAX], front = 0, rear = -1;
int n;

/* BFS function */
void bfs(int start) {
    int i, v;
    visited[start] = 1;
    queue[++rear] = start;

    while (front <= rear) {
        v = queue[front++];
        printf("%d ", v);

        for (i = 0; i < n; i++) {
            if (graph[v][i] == 1 && visited[i] == 0) {
                visited[i] = 1;
                queue[++rear] = i;
            }
        }
    }
}

int main() {
    int i, j, start;

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

    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

    for (i = 0; i < n; i++)
        visited[i] = 0;

    printf("Enter starting vertex: ");
    scanf("%d", &start);

    printf("BFS Traversal: ");
}

```

```
bfs(start);  
return 0;  
}
```

Input

Enter number of vertices: 4

Enter adjacency matrix:

```
0 1 1 0  
1 0 1 1  
1 1 0 0  
0 1 0 0
```

Enter starting vertex: 0

Output

BFS Traversal: 0 1 2 3

```

#include <stdio.h>
#define MAX 10

int graph[MAX][MAX], visited[MAX];
int n;

void dfs(int v) {
    int i;
    visited[v] = 1;

    for (i = 0; i < n; i++) {
        if (graph[v][i] == 1 && visited[i] == 0)
            dfs(i);
    }
}

int main() {
    int i, j;
    int connected = 1;

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

    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

    for (i = 0; i < n; i++)
        visited[i] = 0;

    dfs(0);

    for (i = 0; i < n; i++) {
        if (visited[i] == 0) {
            connected = 0;
            break;
        }
    }

    if (connected)
        printf("Graph is CONNECTED\n");
    else
        printf("Graph is NOT CONNECTED\n");
}

```

```
    return 0;  
}
```

Input

Enter number of vertices: 3

Enter adjacency matrix:

```
0 1 0  
1 0 1  
0 1 0
```

Output

Graph is CONNECTED