```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct Node {
    int vertex;
    struct Node* next;
};

struct Node* adj[MAX];
int visited[MAX];

void addEdge(int u, int v) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->vertex = v;
    newNode->next = adj[u];
    adj[u] = newNode;

    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->vertex = u;
    newNode->next = adj[v];
    adj[v] = newNode;
}

void DFS(int node) {
    visited[node] = 1;

    struct Node* temp = adj[node];
    while (temp != NULL) {
        if (!visited[temp->vertex]) {
            DFS(temp->vertex);
        }
        temp = temp->next;
    }
}

int main() {
    int n, e, u, v;

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

    printf("Enter number of edges: ");
    scanf("%d", &e);
```
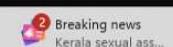
```c
42
43      printf("Enter number of edges: ");
44      scanf("%d", &e);
45
46      for (int i = 0; i < n; i++)
47          adj[i] = NULL;
48
49      printf("Enter edges (u v):\n");
50      for (int i = 0; i < e; i++) {
51          scanf("%d %d", &u, &v);
52          addEdge(u-1, v-1);
53      }
54
55      for (int i = 0; i < n; i++)
56          visited[i] = 0;
57
58      DFS(0);
59      int connected = 1;
60      for (int i = 0; i < n; i++) {
61          if (!visited[i]) {
62              connected = 0;
63              break;
64          }
65      }
66
67      if (connected)
68          printf("The graph is CONNECTED.\n");
69      else
70          printf("The graph is NOT connected.\n");
71
72      return 0;
73  }
74
75
```

```
Enter number of vertices: 6
Enter number of edges: 8
Enter edges (u v):
1 2
1 3
2 6
3 4
3 6
4 5
4 6
5 6
The graph is CONNECTED.

Process returned 0 (0x0)   execution time : 11.526 s
Press any key to continue.
```