```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 50

int graph[SIZE][SIZE] = {{0}, {0}};
int vertice, edges;
int top = -1, capacity = SIZE, stack_arr[SIZE], dfs_arr[SIZE];

int isFull_stack()
{
    if (top == capacity - 1)
        return 1;
    else
        return 0;
}

int isEmpty_stack()
{
    if (top == -1)
        return 1;
    else
        return 0;
}

void push(int item)
{
    if (!isFull_stack())
    {
        top++;
        stack_arr[top] = item;
    }
}

int pop()
{
    int item;
    if (!isEmpty_stack())
    {
        item = stack_arr[top];
        top--;
        return item;
    }
    return -1;
}

int notVisited(int item, int index)
{
    int fact = 1;
    for (int i = 0; i < index; i++)
    {
        if (dfs_arr[i] == item)
        {
            fact = 0;
            break;
        }
```

```
    }
    for (int i = 0; i < top; i++)
    {
        if (stack_arr[i] == item)
        {
            fact = 0;
            break;
        }
    }
    if (fact == 0)
        return 0;
    return 1;
}

int dfs(int ver)
{
    int k = 0;
    push(ver);
    while (!isEmpty_stack())
    {
        int curr_vertex = pop();
        for (int i = 0; i < vertice; i++)
        {
            if (graph[curr_vertex][i] && notVisited(i, k))
            {
                push(i);
            }
        }
        dfs_arr[k] = curr_vertex;
        k++;
    }
    return k;
}

int main()
{
    printf("Enter the number of vertices:");
    scanf("%d", &vertice);
    printf("Enter the number of edges:");
    scanf("%d", &edges);
    srand(time(NULL));
    for (int i = 0; i < edges; i++)
    {
        int r = rand() % vertice;
        int c = rand() % vertice;
        if (r != c)
        {
            int e = 1;
            graph[r][c] = e;
            graph[c][r] = e;
        }
        else
        {

            graph[r][c] = 0;
        }
    }
```

```
    for (int i = 0; i < vertice; i++)
    {
        for (int j = 0; j < vertice; j++)
        {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }

    int ver;
    printf("Enter the vertex at which you want to start finding
spanning tree:");
    scanf("%d", &ver);
    int index = dfs(ver);
    for (int i = 0; i < index; i++)
        printf("%d ", dfs_arr[i]);

    printf("\n Vertice starts at 0.\n");

    return 0;
}
```