Day 84 coding Statement :

Given an undirected graph and an integer M. The task is to determine if the graph can be colored with at most M colors such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means the assignment of colors to all vertices. Print 1 if it is possible to colour vertices and 0 otherwise.

Example 1:

Input:
N = 4
M = 3
E = 5
Edges[] = {(0,1),(1,2),(2,3),(3,0),(0,2)}

Output: 1
Explanation: It is possible to colour the given graph using 3 colours.

Example 2:

Input:
N = 3
M = 2
E = 3
Edges[] = {(0,1),(1,2),(0,2)}
Output: 0

```
import java.util.*;

public class Program {
	public boolean isPossible(boolean[][] graph, int[] color, int node, int col,
int n) {
		for (int i = 0; i < n; i++) {
			if (graph[node][i] && color[i] == col)
				return false;
		}
		return true;
	}

	public boolean solve(int node, boolean[][] graph, int[] color, int m, int n) {
		if (node == n)
			return true;
		for (int i = 1; i <= m; i++) {
			if (isPossible(graph, color, node, i, n)) {
				color[node] = i;
				if (solve(node + 1, graph, color, m, n))
					return true;
				color[node] = 0;
			}
		}
		return false;
```

```
	}

	public boolean graphColoring(boolean graph[][], int m, int n) {
		int[] color = new int[n];
		return solve(0, graph, color, m, n);
	}

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int m = sc.nextInt();
		int e = sc.nextInt();
		int a, b;
		boolean graph[][] = new boolean[n][n];
		for (int i = 1; i <= e; i++) {
			a = sc.nextInt();
			b = sc.nextInt();
			graph[a][b] = true;
			graph[b][a] = true;
		}
		Program obj = new Program();
		if (obj.graphColoring(graph, m, n))
			System.out.print("1");
		else
			System.out.print("0");
	}
}
```