Background
This assignment is based on a problem on the Kattis programming problem website (open.kattis.com). It is a natural problem for a constraint satisfaction framework, and the purpose of this assignment is to get you a bit of practice in this different programming paradigm. You will be using the Z3 theorem prover developed by Microsoft Research. You can run Z3 in Python on the Linux lab machines by simply importing the z3 library. A tutorial (of sorts) on Z3 for Python is located here (http://www.cs.tau.ac.il/~msagiv/courses/asv/z3py/guide-examples.htm) The examples at the bottom of the page may be particularly useful. Run the code snippets as you read to get a feel for how things work.

Problem Description
(from https://open.kattis.com/problems/committeeassignment)
You are the chair of a large department, and, as such, it’s your job to assign department members to committees. Everyone has to be on exactly one committee, but some department members don’t get along and cannot be assigned to the same committee. Of course, you would like to have as few committees as possible since it’s hard to make up jobs for committees you don’t really need (e.g., committee for snack machine pricing oversight, committee for shredding procurement, committee for committee name selection). What you need is a program that figures out how many committees are needed so that no mutually hostile department members are placed on the same committee.

Input and Output
Input consists of department descriptions. Each begins with a positive integer \( n \leq 100 \) and a non-negative integer \( m \leq \frac{n(n-1)}{2} \). The value of \( n \) indicates the number of people there are in the department, and \( m \) tells how many pairs of people don’t get along. These two values should appear on one line, separated by a space. This is followed by \( m \) lines, each listing the combinations of two different department members who don’t get along. This problem specification differs from the Kattis problem input in that it uses integers (1 through \( n \)) to indicate the faculty members, rather than using strings.

For each department description, print out a line showing the minimum number of committees needed to keep mutually hostile department members on different committees or -1 if there is no solution.

Department descriptions should continue to be read and solved until a pair of zeroes (“0 0”) is entered for \( m \) and \( n \). The program should then exit.

Sample Input:
5 0
5 2
Hints

To keep track of assignments of people to committees, use a Function from Int (people) to Int (committees).

To programmatically see if there is a solution to constraints, you need to use the Solver API, not the solve function. For example:

```python
s = z3.Solver()
s.push()
s.add(constraints)
if s.check() == z3.sat:
    print "Satisfiable!"
    print "Here is the model: ", s.model()
    s.pop()
```

Solve the problem by testing committees of increasing size. Start by seeing if the problem can be solved using one committee. Constrain all committee numbers to be between 1 and 1. If this is unsolvable (unsat), try testing with two committees. Keep increasing the number of committees until the problem becomes satisfiable or you know that there is no possible solution.

Submission

Submit your program to autolab as a single .py file that can be run from the command line. You should not have any prompts, the program should just start reading the expected input (autolab will be comparing outputs and the prompts add too much variability).