The Modern Dress Code

Byteland is making preparations for the approaching period of *feasts, cabaret shows, and general merrymaking*. The women in Byteland are especially excited about the Grand Ball to be held on the last Saturday of the holiday period, and have already started making preparations. At present, one of the most important questions for every lady seems to be: *What should I wear for the ball?*. This problem is far more important than you might imagine possible, since the months after the period of feasts will offer no celebrations whatsoever, and during this time every lady likes to think back to how charming and original she looked at the ball. And of course, you have to remember that if two female friends come to the party dressed in the same dresses they will both be the subject of jokes and chuckled remarks for years.

As in most difficult problems among ladies, the gentlemen of Byteland are forcibly involved and asked to help out. So, every day each woman asks her man for advice on what to wear. Unfortunately very few men in Byteland know anything at all about fashion so they all come to you and ask for help.

You do not have a lot of to do with Bytelandian fashion either but, as a computer specialist and a generally clever person who never gives up, you know how to address any problem. You have modeled the situation as follows:

- there are 1<=k<=20 days left till the grand feast,
- dresses in Byteland come in different colors, which you have numbered with integers starting from 0,
- the relationships between Bytelandian women are given by a graph (adjacent nodes denote two friends),
- every lady would like to have a dress of different color than all her friends, otherwise she will be most unhappy.

You have a plan to provide the men with a simple set of rules to help them determine the best, most unique color of clothes for their ladies. Since you only have a limited amount of time to spare, each man will have to content himself with the same algorithm from you. Because most of the men don't use a computer, you should use a very simple language to express your algorithm.

Task

Prepare an algorithm which, if applied properly by all the men, will guarantee that as few of the ladies as possible are unhappy due to their dresses.

A short description of the language you should use is given below:

- each program is a list of rules,
- each < rule > is of the form:

  < rule >::= if < comp_condition > then { < comp_command > };

  < comp_command >::=< command >;
  < comp_command >::=< command >; < comp_command >

< comp_condition >::=< condition >
< comp_condition >::=not < comp_condition >
< comp_condition >::=(< comp_condition > or < comp_condition >)
< comp_condition >::=(< comp_condition > and < comp_condition >)

Please note that after each < command > and < rule > you must put a semicolon.

- < command > is simply an assignment of the form:

  < command >::= < symbol >:=< expr >

- < symbol > can be one of the following strings:
  - a state component: color, a, b,
  - an additional local variable: l0, l1, l2, l3, l4, l5, l6, l7, l8, l9.
- The expression < expr > should be constructed using:
  - writable symbols (as given above) and several special read-only variables which give you some information about the graph:

    deg degree of current vertex
    delta maximum vertex degree in a graph
    n number of vertices
    m number of edges
    nr number of possible colors
    daysleft number of days remaining till the ball

  - operators (in order of precedence):

    + - arithmetic plus and minus
    * / % arithmetic multiplication, division and remainder

  - function rnd(< expr >), returning a random number between 0 and < expr > - 1 inclusive.
  - functions mina, minb, minc returning the minimal value of variable a, b, color respectively, taken over all the neighbours of the vertex (or the smallest integer for a vertex without neighbours). Functions maxa, maxb, maxc act similarly for maximum values of variables.
- < condition > is a logical expression taking one of two forms:

  < condition >::= < expr > < operator > < expr >
  < condition >::= < exist-operator > (< expr >)

  The binary < operator > is one of the comparison operators ==, <, or >. The unary < exist-operator > is one of the following functions: Eaeq, Ebeq, Eceq, which return true iff for some neighbour of the given vertex the value of variable a, b, color, respectively, is equal to < expr >.

The algorithm has to be applied by every man every day. More formally speaking: at the start of the process all variables have random values. Every morning all the men assign 0 to their variables l0,l1,...,l9. Exactly at noon, each lady comes to ask for advice. The man does all he can to help her: he processes all his rules from top to bottom, and repeats the process as long as at least one of the IF clauses is true. (However, if he has to do it more than one hundred times, he will give up in dispair). Then he tells the lady the color he has chosen for her. Every evening, the men meet at the pub, and when they have nothing left to talk about, they exchange information about how the women are boring them, and what values of a, b and color they have had to come up with today. (It is interesting to note that, as a side effect, functions E_eq, min_ and max_ actually make use of the previous day's values of the neighbours' variables.)

Finally, on the day of the ball the women put on clothing corresponding to the color last suggested by their men and it is time to judge how effective your algorithm has been. (This is one of these tasks which, if done badly, may result in getting the programmer lynched by an angry mob.)

Score

There are 50 test cases on which your program will be tested. Graphs used in tests will have no more than 25 vertices. Your score is the sum of scores of your program taken over all test cases. For each test case the score is equal to the ratio of the number of correctly colored vertices (i.e. such that 0<= color< nr and Eceq(color) is false) to the total number of vertices. If all vertices are correctly colored, a bonus of 1/(1+ *maximum color used*) points is awarded.

A program will be assessed as Compilation Error if it cannot be interpreted due to syntactic errors. If on a given day the rules for a vertex cannot be processed within 100 iterations or the entire simulation takes more than about 60s, your program will be judged as Time Limit Exceeded.

Example

Submit code in the language TEXT, the judge will interpret it properly. Here is an example of a simple single-rule program.

Code:
```
if (l0==0 and not Eceq ((color-1)%n)) then {color:=(color-1)%n; l0:=1;};
```

Score:
5.491