1. Table 5 seconds 20 points

You are given a table of numbers with R rows and V columns. It is known that the items in each row of the table are ordered non-decreasingly.

Write a program to determine whether there exists a value that apperas in every row of the table.

Input. On the first line of the text file `TBL.SIS` are two integers, R and V ($1 \leqslant R \leqslant 500$, $1 \leqslant V \leqslant 500$), separated by a space — the number of rows and the number columns of the table. On each of the next R lines are V integers, separated by spaces — contents of the table, listed line by line, from top to bottom and left to right. Absolute values of the items do not exceed 30 000.

Output. The first line of the text file `TBL.VAL` should contain the word `JAH`, if there exists a value that appears at least once in every row of the table, or the word `EI`, if there is no such value. If the value exists, the second line of the file should contain it. If more than one value appears in every row of the table, output any one of them.

Example.

```
TBL.SIS      TBL.VAL
2 3          JAH
1 2 3        3
3 4 5
```

Example.

```
TBL.SIS      TBL.VAL
2 3          EI
1 2 3
4 5 6
```

Grading. Points for negative test cases are awarded only to programs that solve correctly at least one positive test case.

2. Introduction 5 seconds 40 points

There are N guests invited to a party, and some of them already know each other. Of course, the relationship is symmetric (that is, if A knows B then B knows A).

Write a program to divide the guests into groups for a game of introduction. The objective is to create as few groups as possible under the condition that no two quests who already know each other can be in the same group.

Input. On the first line of the text file `TUT.SIS` are two integers, N and K ($1 \leqslant N \leqslant 100$, $0 \leqslant K \leqslant 100$), separated by a space, where N is the number of guests and K is the number of explicitly given facts of acquaintance. All guests are labelled $1 \ldots N$. Each of the following K lines contain two integers, A_i and B_i ($1 \leqslant A_i \leqslant N$, $1 \leqslant B_i \leqslant N$), separated by a space, meaning that the guests A_i and B_i know each other.

Output. The first line of the text file `TUT.VAL` should contain an integer M, the number of groups created. Let's label the groups $1 \ldots M$. The second line of the file should contain M integers, separated by spaces — label of the group for each guest, in the order of labels of guests. If there are multiple solutions with the same number of groups, any one of them could be listed.

Example.

```
TUT.SIS      TUT.VAL
5 3          2
1 2          1 2 1 2 2
3 4
5 1
```

Remark. In the example output, the groups are $\{1, 3\}$ and $\{2, 4, 5\}$. Another possible solution would be $\{1, 4\}$ and $\{2, 3, 5\}$.

3. The `crontab` file open tests 40 points

In a computer system, it is often necessary to perform some actions according to a given schedule. In Unix-like operating systems, this service is offered by a program called `crond` (Greek *chronos* 'time' and *dæmon* 'spirit') that reads the schedule from a file called `crontab`.

Each line of the `crontab` file consists of six fields separated from each other by spaces or tabs. Five first fields (which may not contain spaces or tabs) describe the schedule of performing an action and the last one contains the command to be executed for performing this action.

The fields that describe the schedule are: minutes (0 . . . 59), hours (0 . . . 23), dates (1 . . . 31), months (1 . . . 12), and weekdays (0 . . . 6, where 0 = Sunday, 1 = Monday, . . . , 6 = Saturday). Each field may contain a star (`*`), which means all valid values, or a comma-separated list. In a list, each item may be either a valid value or two values separated by a dash (`-`), meaning all valid values from the first value to the second one (the bondary values themselves included).

Generally, the command is executed at times satisfying all the given conditions. For example, if the minutes field contains `0,30` and hours field contains `12-14`, the command is executed at 12:00, 12:30, 13:00, 13:30, 14:00, and 14:30. The date and weekday fields are an exception. If both of these fields contain a non-star, the command is executed at times meeting either of the conditions. For example, if the date field contains `10,20` and the weekday field contains `5`, the command is executed on the 10th and 20th day of month (regardless of the weekday) and every Friday (regardless of the date).

You are given a list of commands and the times when they should be executed. The task is to create a `crontab` file scheduling all the given commands for execution at specified times with as few lines as possible. If several commands are to be executed at the same time, they are really executed in the order of their entries in the `crontab` file. Your reconstructed `crontab` should preserve that order.

Input. Each line of the text file `CRON.SIS` contains three space-separated fields: the date in the form DD.MM.YYYY, the time in the form HH:MM, and the command to be executed on this date at this time. The file is complete: it lists all actions to be performed from the moment specified on the first line to the moment specified on the last line. The lines in the file are given in chronological order.

Output. The text file `CRON.VAL` should contain the `crontab` to make `crond` to execute exactly the desired commands in the given timeframe. What would happen before the first or after the last moment specified in the input file does not matter.

Example.

```
CRON.SIS
20.10.2003 12:00 test
20.10.2003 13:00 test
20.10.2003 14:00 test
20.10.2003 17:00 test
21.10.2003 12:00 test
```

```
CRON.VAL
0 12-14,17 * * * test
```

Remark. The `crontab` given in the output file executes the command `test` every day at 12:00, 13:00, 14:00, and 17:00. Thus the list of actions performed from 12:00 on 20.10.2003 to 12:00 on 21.10.2003 matches the list given in the input file. Therefore, the given `crontab` meets the requirements, even though the command `test` is also executed before 20.10.2003 and after 12:00 on 21.10.2003, in particular at 13:00, 14:00, and 17:00 on 21.10.2003.

Grading. In this task, you are given 10 specific input files named `cron01.sis` to `cron10.sis` and you should, as your solution, submit the corresponding output files named `cron01.val` to `cron10.val`. It is not necessary to submit a program. Every correct output file receives points inversely proportional to the ratio between the number of lines in this file and the best solution submitted (that is, a file with twice the number of lines receives half the points).