1. Walk (`jalu`)

1 second 10 points

Mart takes a walk every day and his sports watch registers the duration and speed of each segment of uniform movement of his walk. After the walk, Mart can download a log file to his computer. Find the total distance and the average speed of the walk based on this file.

Input. The first line of the text file `jalusis.txt` contains an integer N ($1 \le N \le 10{,}000$) and each of the subsequent N lines contains the following (space-separated) data about one segment:

- the duration of the segment in minutes and seconds in the form M`m`S`s`, where M is the integer number of minutes (and the minutes part is omitted for times less than a minute) and S is the integer number of seconds (which never exceeds 59);
- the word `kiirusega`;
- the speed on this segment expressed as minutes and seconds per kilometre in the form M`m`S`s/km`, where M is the integer number of minutes and S is the integer number of seconds (which never exceeds 59), and you may assume Mart needs at least a minute, but less than an hour, per kilometre.

You may also assume that the total of the durations of all segments does not exceed 24 hours.

Output. The text file `jaluval.txt` should contain two lines:

- on the first line the total distance of the walk, rounded to metres, in the form L`m`, where L is the integer number of metres;
- on the second line the average speed in the form V`km/h`, where V is a real number; the output value must not differ from the correct answer by more than 0.001.

Example.

`jalusis.txt`
```
3
8m30s kiirusega 9m10s/km
4m10s kiirusega 8m1s/km
21s kiirusega 4m10s/km
```

`jaluval.txt`
```
1531m
7.057km/h
```

2. Carpets (vaip)

1 second 20 points

The fanciest hall of Prince Ville's new castle has a rectangular floor. The walls of the hall run from North to South and from East to West. Ville's advisors have proposed to cover some parts of the floor with carpets and even suggested a specific arrangement of them.

Ville, caring mostly only about the quantitative aspect of the deal, would like to know the total area covered by the carpets in the proposed arrangement.

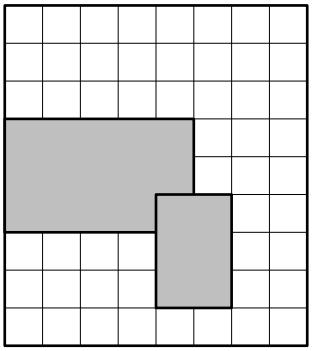
Input. The first line of the text file `vaipsis.txt` contains three space-separated integers: the number of carpets K ($1 \le K \le 3$), the distance L from the East wall to the West wall ($1 \le L \le 10^6$), and the distance P from the North wall to the South wall ($1 \le P \le 10^6$) of the hall. Each of the following K lines contains four space-separated integers N, S, E, and W describing one carpet. The edges of the carpet are parallel to the walls and the numbers N, S, E, and W show, respectively, the distance of the North, South, East, and West edge of the carpet from the corresponding wall. The carpets may overlap partially or fully. All distances are given in metres.

Output. The only line of the text file `vaipval.txt` should contain a single integer S, the total area of the floor covered by carpets, in square metres.

Example.

`vaipsis.txt`
```
2 8 9
3 3 3 0
5 1 2 4
```

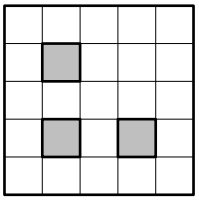
`vaipval.txt`
```
20
```



Example.

`vaipsis.txt`
```
3 5 5
3 1 1 3
1 3 3 1
3 1 3 1
```

`vaipval.txt`
```
3
```



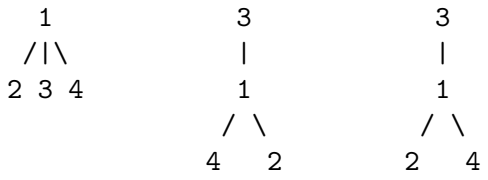
3. Turning a Tree (puu)

1 second 30 points

You are given a tree with nodes numbered $1 \ldots N$, where node 1 is the root of the tree and for each node the list of its child nodes is known.[1]

Find the tree we would get by lifting the leaf K of the original tree to be the new root, but leaving all edges intact, including the relative ordering of the edges at each node.

For example, starting from the tree shown on the left in the figure below and making the leaf 3 the new root, we would get the tree shown in the middle in the figure. The three shown on the right in the figure would not be correct answer, because the neighbors for the node 1 (listed counter-clockwise) are 2, 3, 4 in the original tree, but 2, 4, 3 in this tree.



Input. The first line of the text file `puusis.txt` contains the number of nodes N ($1 \le N \le 10{,}000$) and the index K of the leaf to become the new root ($1 \le K \le N$). The following N lines describe the structure of the original tree. The $(i+1)$-th line first contains m_i, the number of child nodes of the node i, and then the indices of the m_i child nodes, listed from left to right.

Output. The text file `puuval.txt` should contain exactly N lines: the structure of the new tree, in the format used in the input file.

Example.

puusis.txt	puuval.txt
4 3	2 4 2
3 2 3 4	0
0	1 1
0	0
0	

Explanation of the output lines:

1. Node 1 has 2 children, nodes 4 and 2 (in this order).
2. Node 2 has no children.
3. Node 3 has 1 child, node 1.
4. Node 4 has no children.

Grading. In test cases worth 16 points in total, the input is a binary tree (no node of the original tree has more than 2 children).

[1] See also `http://en.wikipedia.org/wiki/Tree_(data_structure)`

4. Paper Strips (`riba`)

6/30 seconds 40 points

Jack has an L cm long black strip of paper. He will glue colored strips of paper onto the black one. It may be assumed the new strips will always fit on top of the black one. All the strips have the same width. Each new strip will hide the colors under it.

Find the colors and lengths of the segments visible in the end.

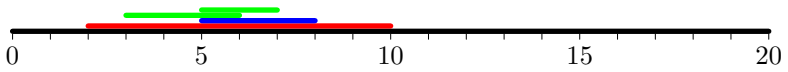
Input. The first line of the text file `ribasis.txt` contains the length of the black strip L and the number of the colored strips N. Each of the following N lines contains three integers describing one colored strip: the color code K ($1 \le K \le 100$) and the distance A of the beginning and the distance B of the end of the colored strip from the beginning of the black strip ($0 \le A < B \le L$). The code of color black is 0. Only the initial strip is black, all others are colored.

Output. The text file `ribaval.txt` should contain the colors and lengths of the segments of each color in the final result, listed from the beginning of the original black strip to its end. Adjacent strip sections of the same color should be output as one segment.

Example.

`ribasis.txt`
```
20 4
1 2 10
2 5 8
3 3 6
3 5 7
```

`ribaval.txt`
```
0 2
1 1
3 4
2 1
1 2
0 10
```



Grading. In all test cases $1 \le L \le 10^9$ and $0 \le N \le 500{,}000$. In test cases worth 20 points in total, additionally $0 \le N \le 1{,}000$, and among them in test cases worth 10 points in total, furthermore $1 \le L \le 1{,}000$.

5. Dorm Party (pidu)

15/120 seconds 50 points

In a regular dorm, two stundents share a room: the freshman Jack and the party animal Jude. One day Jude got the idea of hosting a party in their room. Jack, on the other hand, likes peace and quiet and these are hard to come by in a dorm as it is. Since Jack also knows that Jude has a lot of friends, the thought of the party in their 18 m^2 room makes him shiver.

Jude has decided to invite N girls and M boys to the party. Each girl is interested in some (possibly empty) set of boys and the same set of boys is interested in the girl. To make the party more lively, Jude wants to arrange pairs so that the largest possible number of guests would dance. He can't, however, make a pair dance unless they are interested in each other. Jude, being ignorant in sciences and having no clue what maximum flow even means, got stuck and asked Jack for help.

Jack is afraid that the party might be a success and inspire Jude to organize more of them. It is well known that the level of noise is the main indicator of a party's success. First, a dancing guest is louder than a non-dancing one. Second, if there are two not yet dancing people at the party who are interested in each other, they will spontaneously form a new pair and start dancing; being proud of their initiative (and also somewhat drunk), they will be extra noisy. Jack wants to avoid such pairs at all cost, and furthermore present Jude with a pairing plan that would keep the noise minimal (the least possible number of dancing pairs with no additional pairs of people interested in each other). After thinking for a while, he realized that, given the fairly large crowd Jude has invited, this is not trivial.

Find an arrangement suitable for Jack.

Input. The first line of the text file `pidusis.txt` contains the number of girls N ($1 \leq N \leq 19$), the number of boys M ($1 \leq M \leq 19$), and the number of mutually interested pairs K ($0 \leq K \leq N \cdot M$). Each of the following K lines contains two integers A_i ($1 \leq A_i \leq N$) and B_i ($1 \leq B_i \leq M$) meaning that the girl A_i and the boy B_i are interested in each other.

Output. The first line of the text file `piduval.txt` should contain an integer S, the number of dancing pairs in the arrangement. Each of the following S lines should contain two space-separated integers X_i and Y_i meaning that the girl X_i and the boy Y_i should form a pair.

Example.

`pidusis.txt`

```
6 5 9
1 1
2 1
3 2
4 3
5 2
5 3
5 4
6 3
6 5
```

`piduval.txt`

```
3
2 1
5 2
6 3
```

Grading. In this task the test cases are divided into groups and a solution will receive points for a group only if it solves all test cases in the group. The following additional conditions hold in the groups:

1. Any arrangement that does not induce an extra noisy pair also has the minimal number of pairs possible for such arrangements (10 points)
2. $K \leq 24$ (10 points)
3. $N + M \leq 22$ (20 points)
4. No additional conditions (10 points)

Remark. For solving this problem, it could be useful to learn about the maximal matching and the maximum flow in a graph:
`http://en.wikipedia.org/wiki/Matching_(graph_theory)`
`http://en.wikipedia.org/wiki/Maximum_flow_problem`

6. Skyscrapers (`torn`)

50 points

Each cell of an $N \times N$ grid contains a skyscraper. The height of each tower is in the range $1 \ldots N$ and there are no two towers of equal height in any row or any column.

For each row and each column, we know how many skyscrapers are visible when looking from the end of it (the higher ones nearer to the viewer hide the lower ones farther from the viewer).

Find a possible arrangement of the skyscraprers.

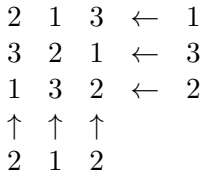
Input. The first line of the input file contains the integer N. The following N lines contain the numbers of towers visible from the end of each row (from top to bottom), followed by N lines containing the numbers of towers visible from the end of each column (from left to right).

Output. The output file should contain exactly N lines. Each line should contain exactly N integers, the heights of the towers. If there are several possible solutions, output any one of them.

Example.

Input file	Output file
3	2 1 3
1	3 2 1
3	1 3 2
2	
2	
1	
2	

The figure below shows the connection between the input and output: for example, a viewer at the end of the first row would see only one tower, as the highest one would be closest to him and hide all others; a viewer at the end of the last column, on the other hand, would see two towers, because the tower with height 2 would hide the tower with height 1, but the tower with height 3 would still be visible.



Grading. In this task you are given (via the grading server) 10 input files and you should submit the corresponding output files as your solution. You should not submit a program and it will not be graded.

7. Rock-Paper-Scissors (`mang`)

60 seconds 50 points

The rock-paper-scissors game is played as follows: Two players simultaneously pick their moves as 'rock', 'paper', or 'scissors'. When both pick the same move, they each get zero points. When they pick different moves, the rock beats the scissors, the scissors beat the paper, and the paper beats the rock. The player who picked the winning move gets $+1$ and the loser -1 points. A match consists of a number of moves whose poinst are added. Thus, the sum of points of the two players at the end of a match will always be zero.

There are many rock-paper-scissors competitions where the main goal is to guess the opponent's strategy and play against it.

Task. The jury has implemented 10 simple programs that play rock-paper-scissors. Write a program to play against them and score as many points as possible.

Match. Your program is started anew for each opponent. Your program should write its move, followed by a newline, to the standard output and then read the opponent's move from the standard input. Each move is designated by a capital letter: `K` (rock), `P` (paper), or `R` (scissors). The match is managed by a judge program that gets the moves of the jury's program in the same way and keeps track of the score. The end of the match is signaled by `L` as the opponent's move and after that your program should terminate immediately.

Time limit. Each match lasts for 100,000 moves or for 1 minute, whichever limit is reach first. Thus, your program has a chance to score more by playing faster. You may assume that the jury's program runs very fast. You may also assume that the strategy of the jury's program can be deduced with reasonbale effort.

Grading. The scores of all matches of your program are added. If the total is negative, your score for this task is zero. The contestant with the best result gets 50 points, other contestants with positive results will score proportionally.