

Problem A

Best Relay Team

Problem ID: bestrelayteam
Time limit: 1 second

You are the coach of the national athletics team and need to select which sprinters should represent your country in the 4×100 m relay in the upcoming championships.

As the name of the event implies, such a sprint relay consist of 4 legs, 100 meters each. One would think that the best team would simply consist of the 4 fastest 100 m runners in the nation, but there is an important detail to take into account: flying start. In the 2nd, 3rd and 4th leg, the runner is already running when the baton is handed over. This means that some runners – those that have a slow acceleration phase – can perform relatively better in a relay if they are on the 2nd, 3rd or 4th leg.



Picture by Fernando Frazão/Agência Brasil, cc by

You have a pool of runners to choose from. Given how fast each runner in the pool is, decide which four runners should represent your national team and which leg they should run. You are given two times for each runner – the time the runner would run the 1st leg, and the time the runner would run any of the other legs. A runner in a team can only run one leg.

Input

The first line of input contains an integer n , the number of runners to choose from ($4 \leq n \leq 500$). Then follow n lines describing the runners. The i 'th of these lines contains the name of the i 'th runner, the time a_i for the runner to run the 1st leg, and the time b_i for the runner to run any of the other legs ($8 \leq b_i \leq a_i < 20$). The names consist of between 2 and 20 (inclusive) uppercase letters 'A'-'Z', and no two runners have the same name. The times are given in seconds with exactly two digits after the decimal point.

Output

First, output a line containing the time of the best team. The precise formatting of the time is not important. Then output four lines containing the names of the runners in that team. The first of these lines should contain the runner you have picked for the 1st leg, the second line the runner you have picked for the 2nd leg, and so on. Any solution that results in the fastest team is acceptable.

Sample Input 1

```
6
ASHMEADE 9.90 8.85
BLAKE 9.69 8.72
BOLT 9.58 8.43
CARTER 9.78 8.93
FRATER 9.88 8.92
POWELL 9.72 8.61
```

Sample Output 1

```
35.54
CARTER
BOLT
POWELL
BLAKE
```

Problem B

Galactic Collegiate Programming Contest

Problem ID: gcpc
Time limit: 6 seconds

One hundred years from now, in 2117, the International Collegiate Programming Contest (of which the NCPC is a part) has expanded significantly and it is now the Galactic Collegiate Programming Contest (GCPC).

This year there are n teams in the contest. The teams are numbered $1, 2, \dots, n$, and your favorite team has number 1.

Like today, the score of a team is a pair of integers (a, b) where a is the number of solved problems and b is the total penalty of that team.

When a team solves a problem there is some associated penalty (not necessarily calculated in the same way as in the NCPC – the precise details are not important in this problem). The total penalty of a team is the sum of the penalties for the solved problems of the team.

Consider two teams t_1 and t_2 whose scores are (a_1, b_1) and (a_2, b_2) . The score of team t_1 is better than that of t_2 if either $a_1 > a_2$, or if $a_1 = a_2$ and $b_1 < b_2$. The rank of a team is $k + 1$ where k is the number of teams whose score is better.

You would like to follow the performance of your favorite team. Unfortunately, the organizers of GCPC do not provide a scoreboard. Instead, they send a message immediately whenever a team solves a problem.



Picture by GuillaumePreat on Pixabay, cc0

Input

The first line of input contains two integers n and m , where $1 \leq n \leq 10^5$ is the number of teams, and $1 \leq m \leq 10^5$ is the number of events.

Then follow m lines that describe the events. Each line contains two integers t and p ($1 \leq t \leq n$ and $1 \leq p \leq 1000$), meaning that team t has solved a problem with penalty p . The events are ordered by the time when they happen.

Output

Output m lines. On the i 'th line, output the rank of your favorite team after the first i events have happened.

Sample Input 1

```
3 4
2 7
3 5
1 6
1 9
```

Sample Output 1

```
2
3
2
1
```

3 4	2
2 7	3
3 5	2
1 6	1
1 9	

Problem C

Distinctive Character

Problem ID: distinctivecharacter
Time limit: 4 seconds

Tira would like to join a multiplayer game with n other players. Each player has a character with some features. There are a total of k features, and each character has some subset of them.

The similarity between two characters A and B is calculated as follows: for each feature f , if both A and B have feature f or if none of them have feature f , the similarity increases by one.

Tira does not have a character yet. She would like to create a new, very original character so that the maximum similarity between Tira's character and any other character is as low as possible.

Given the characters of the other players, your task is to create a character for Tira that fulfils the above requirement. If there are many possible characters, you can choose any of them.



Picture by Fairytalemaker on Pixabay

Input

The first line of input contains two integers n and k , where $1 \leq n \leq 10^5$ is the number of players (excluding Tira) and $1 \leq k \leq 20$ is the number of features.

Then follow n lines describing the existing characters. Each of these n lines contains a string of k digits which are either 0 or 1. A 1 in position j means the character has the j 'th feature, and a 0 means that it does not have the j 'th feature.

Output

Output a single line describing the features of Tira's character in the same format as in the input. If there are multiple possible characters with the same smallest maximum similarity, any one of them will be accepted.

Sample Input 1

```
3 5
01001
11100
10111
```

Sample Output 1

```
00010
```

Sample Input 2

```
1 4
0000
```

Sample Output 2

```
1111
```

Problem D

Import Spaghetti

Problem ID: importspaghetti
Time limit: 4 seconds

You just graduated from programming school and nailed a Python programming job. The first day at work you realize that you have inherited a mess. The *spaghetti* design pattern was chosen by the previous maintainer, who recently fled the country. You try to make sense of the code, but immediately discover that different files depend cyclically on each other. Testing the code, in fact running the code, has not yet been attempted.

```
root@programming-conpo:~# python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170118] on linux2
Type "help", "copyright", "credits" or "license()" for more information.
>>> from __future__ import braces
      File "<stdin>", line 1
SyntaxError: not a chance
>>>
```

cc-by NCPC 2017

As you sit down and think, you decide that the first thing to do is to eliminate the cycles in the dependency graph. So you start by finding a shortest dependency cycle.

Input

The first line of input contains a number n , $1 \leq n \leq 500$, the number of files. Then follows one line with n names of files. Each name is a string with at least 1 and at most 8 lower case letters 'a' to 'z'. Then follow n sections, one section per file name, in the order they were given on the second line. Each section starts with one line containing the name of the file and an integer k , followed by k lines, each starting with "import".

Each "import" line is a comma-space separated line of dependencies. No file imports the same file more than once, and every file imported is listed in the second line of the input. Comma-space separated means that every line will start with "import", then have a list of class names separated by ", " (see sample inputs for examples).

Output

If the code base has no cyclic dependencies, output "SHIP IT". Otherwise, output a line containing the names of files in a shortest cycle, in the order of the cycle. If there are many shortest cycles, any one will be accepted.

Sample Input 1

```
4
a b c d
a 1
import d, b, c
b 2
import d
import c
c 1
import c
d 0
```

Sample Output 1

```
c
```

Sample Input 2

```
5
classa classb myfilec execd libe
classa 2
import classb
import myfilec, libe
classb 1
import execd
myfilec 1
import libe
execd 1
import libe
libe 0
```

Sample Output 2

```
SHIP IT
```

Sample Input 3

```
5
classa classb myfilec execd libe
classa 2
import classb
import myfilec, libe
classb 1
import execd
myfilec 1
import libe
execd 1
import libe, classa
libe 0
```

Sample Output 3

```
classa classb execd
```

Problem E

Kayaking Trip

Problem ID: kayaking
Time limit: 2 seconds

You are leading a kayaking trip with a mixed group of participants in the Stockholm archipelago, but as you are about to begin your final stretch back to the mainland you notice a storm on the horizon. You had better paddle as fast as you can to make sure you do not get trapped on one of the islands. Of course, you cannot leave anyone behind, so your speed will be determined by the slowest kayak. Time to start thinking; How should you distribute the participants among the kayaks to maximize your chance of reaching the mainland safely?



Solution to Sample Input 1, with kayaks replaced by canoes (cc by-sa NCPC 2017)

The kayaks are of different types and have different amounts of packing, so some are more easily paddled than others. This is captured by a speed factor c that you have already figured out for each kayak. The final speed v of a kayak, however, is also determined by the strengths s_1 and s_2 of the two people in the kayak, by the relation $v = c(s_1 + s_2)$. In your group you have some beginners with a kayaking strength of s_b , a number of normal participants with strength s_n and some quite experienced strong kayakers with strength s_e .

Input

The first line of input contains three non-negative integers b , n , and e , denoting the number of beginners, normal participants, and experienced kayakers, respectively. The total number of participants, $b + n + e$, will be even, at least 2, and no more than 100 000. This is followed by a line with three integers s_b , s_n , and s_e , giving the strengths of the corresponding participants ($1 \leq s_b < s_n < s_e \leq 1\,000$). The third and final line contains $m = \frac{b+n+e}{2}$ integers c_1, \dots, c_m ($1 \leq c_i \leq 100\,000$ for each i), each giving the speed factor of one kayak.

Output

Output a single integer, the maximum speed that the slowest kayak can get by distributing the participants two in each kayak.

Sample Input 1

```
3 1 0
40 60 90
18 20
```

Sample Output 1

```
1600
```

Sample Input 2

```
7 0 7
5 10 500
1 1 1 1 1 1 1
```

Sample Output 2

```
505
```

Problem F

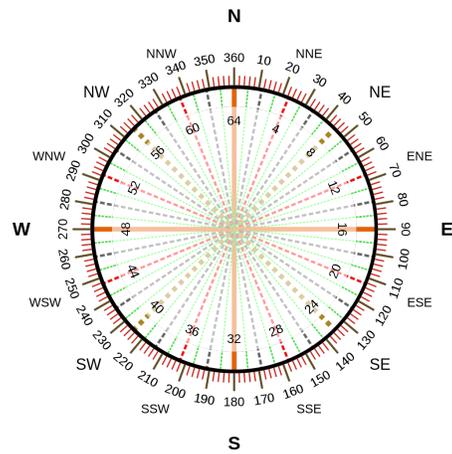
Compass Card Sales

Problem ID: compasscard
Time limit: 6 seconds

Katla has recently stopped playing the collectible card game Compass. As you might remember, Compass is a game where each card has a red, a green and a blue angle, each one between 0 and 359, as well as an ID. Since she has stopped playing, Katla has decided to sell all her cards. However, she wants to keep her deck as unique as possible while selling off the cards. Can you help her figure out the order in which she should sell the cards?

To decide how unique a card is in the deck, she proceeds as follows. For each of the three colors she finds the closest other card in both directions, and then computes the angle between these two other cards. For instance if she has three cards with red angles 42, 90 and 110, then the uniqueness values of their red angles are 340, 68, and 312, respectively. If two cards A and B have the same angle, B is considered the closest to A in both directions so that the uniqueness value of A (and B) for that color is 0.

By summing the uniqueness values over the three colours, Katla finds how unique each card is. When selling a card, Katla sells the currently least unique card (smallest uniqueness value). If two cards have the same uniqueness value, she will sell the one with the higher ID first. After each card is sold, the uniqueness values of the remaining cards are updated before selling the next card.



Input

The first line of input contains an integer n , the number of cards ($1 \leq n \leq 10^5$). Then follows n lines. Each of these n lines contains 4 integers r, g, b, id ($0 \leq r, g, b < 360$, $0 \leq id < 2^{31}$), giving the red, green and blue angles as well as the ID of a card. No two cards have the same ID.

Output

Output n lines, containing the IDs of the cards in the order they are to be sold, from first (least unique) to last (most unique).

Sample Input 1

<pre>3 42 1 1 1 90 1 1 2 110 1 1 3</pre>	<h4>Sample Output 1</h4> <pre>2 3 1</pre>
--	---

Sample Input 2

<pre>4 0 0 0 0 120 120 120 120 240 240 240 240 0 120 240 2017</pre>	<h4>Sample Output 2</h4> <pre>2017 240 120 0</pre>
---	--

Problem G

Hubtown

Problem ID: hubtown
Time limit: 10 seconds

Hubtown is a large Nordic city which is home to n citizens. Every morning, each of its citizens wants to travel to the central hub from which the city gets its name, by using one of the m commuter trains which pass through the city. Each train line is a ray (i.e., a line segment which extends infinitely long in one direction), ending at the central hub, which is located at coordinates $(0, 0)$. However, the train lines have limited capacity (which may vary between train lines), so some train lines may become full, leading to citizens taking their cars instead of commuting. The city council wishes to minimize the number of people who go by car. In order to do this, they will issue instructions stating which citizens are allowed to take which train.

A citizen will always take the train line which is of least angular distance from its house. However, if a citizen is exactly in the middle between two train lines, they are willing to take either of them, and city council can decide which of the two train lines the citizen should use. See Figure H.1 for an example.

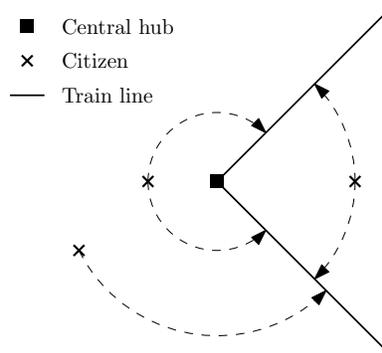


Figure H.1: Illustration of Sample Input 1. The dashed arrows indicate which train lines the citizens are closest to (note that we are measuring angular distances, not Euclidean distance).

Your task is to help the council, by finding a maximum size subset of citizens who can go by train in the morning to the central hub, ensuring that each of the citizens take one of the lines they are closest to, while not exceeding the capacity of any train line. For this subset, you should also print what train they are to take.

Input

The first line of input contains two integers n and m , where $0 \leq n \leq 200\,000$ is the number of citizens, and $1 \leq m \leq 200\,000$ is the number of train lines.

The next n lines each contain two integers x and y , the Cartesian coordinates of a citizen's home. No citizen lives at the central hub of the city.

Then follow m lines, each containing three integers x , y , and c describing a train line, where (x, y) are the coordinates of a single point (distinct from the central hub of the city) which the train line passes through and $0 \leq c \leq n$ is the capacity of the train line. The train line is the ray starting at $(0, 0)$ and passing through (x, y) .

All coordinates x and y (both citizens' homes and the points defining the train lines) are bounded by 1000 in absolute value. No two train lines overlap, but multiple citizens may live at the same coordinates.

Output

First, output a single integer s – the maximum number of citizens who can go by train. Then, output s lines, one for each citizen that goes by train. On each line, output the index of the citizen followed by the index of the train line the citizen takes. The indices should be zero-indexed (i.e., between 0 and $n - 1$ for citizens, and between 0 and $m - 1$ for train lines, respectively), using the same order as they were given in the input.

Sample Input 1

```
3 2
2 0
-1 0
-2 -1
1 -1 1
1 1 2
```

Sample Output 1

```
3
0 1
1 1
2 0
```

Sample Input 2

```
6 3
1 1
1 1
1 1
-1 1
-1 1
0 1
-1 0 2
0 1 2
1 0 2
```

Sample Output 2

```
6
0 2
1 2
2 1
5 1
3 0
4 0
```