# 7012 Kings of Persia

Mahya loves to know more about the history of her country. She is in particular interested in the history of the ancient kings of Persia. Recently, Mahya got curious to know how long each of her favorite kings had lived. So, she started searching the web, and collecting information about the kings lives.

Unfortunately, in most cases, the exact dates on which the kings were born or died are not available in resources. So, Mahya could only find some ranges for possible dates of birth and death for each of the kings. For example, for Cyrus the Great, she could only find that the date of birth was between 600 BC and 575 BC, and the date of death was 530 BC. So, she concluded that Cyrus the Great had lived at least 45 years and at most 70 years.

Mahya has created a long list of her favorite kings, and for each king, has written down two ranges showing the birth range and the death range of that king. Since the list is a bit lengthy, she needs your help to process the list, and produce for each king the minimum and the maximum age.

Note that if a king was born in year $x$ and died in year $y$, then he lived $y - x$ years.

## Input

There are multiple test cases in the input. Each test case consists of a line containing four integers $a$, $b$, $c$, $d$, where $-5000 \leq a \leq b < c \leq d \leq 2000$. The range $[a, b]$ shows the birth range, and $[c, d]$ shows the death range. The input terminates with '`0 0 0 0`' which should not be processed.

## Output

For each test case, output a line containing the minimum and the maximum age as two integers separated by a space.

## Sample Input

```
100 110 180 185
-600 -575 -530 -530
-25 10 72 86
0 0 0 0
```

## Sample Output

```
70 85
45 70
62 111
```

## 7013 Web Colors

Web colors are colors used in displaying web pages. Each color may be specified either as an RGB triple, or a common English name used for that color. Colors are specified according to the intensity of their red, green and blue components, each represented by eight bits. Thus, there are 24 bits used to specify a web color, and totally 16,777,216 colors can be imagined as web colors. But the HTML 4 specification defines only 16 named colors as shown in the table.

| # | Name | Red | Green | Blue |
|---|---|---|---|---|
| 1 | White | 255 | 255 | 255 |
| 2 | Silver | 192 | 192 | 192 |
| 3 | Gray | 128 | 128 | 128 |
| 4 | Black | 0 | 0 | 0 |
| 5 | Red | 255 | 0 | 0 |
| 6 | Maroon | 128 | 0 | 0 |
| 7 | Yellow | 255 | 255 | 0 |
| 8 | Olive | 128 | 128 | 0 |
| 9 | Lime | 0 | 255 | 0 |
| 10 | Green | 0 | 128 | 0 |
| 11 | Aqua | 0 | 255 | 255 |
| 12 | Teal | 0 | 128 | 128 |
| 13 | Blue | 0 | 0 | 255 |
| 14 | Navy | 0 | 0 | 128 |
| 15 | Fuchsia | 255 | 0 | 255 |
| 16 | Purple | 128 | 0 | 128 |

It is often useful to map one given color to one of the HTML named colors. The goal of this problem is to perform just such a mapping in the RGB color space. The input to the program consists of a collection of RGB color values to be mapped to the closest HTML named color.

For a given color, the "closest" color in the HTML color names is a color with the smallest Euclidean distance from the given color. That is, if $rgb$ is the color to be mapped, and $\{R_1G_1B_1, \ldots, R_{16}G_{16}B_{16}\}$ is the set of the HTML colors, the closest color is the one which minimizes the distance expression

$$d = \sqrt{(R_i - r)^2 + (G_i - g)^2 + (B_i - b)^2}$$

where $i$ is an integer from 1 to 16.

## Input

There are multiple test cases in the input. Each test case consists of a line containing three integers $0 \le r, g, b \le 255$ which are the Red, Green and Blue intensities of the color, respectively. The input terminates with '`-1 -1 -1`' which should not be processed.

## Output

For each test case, output a line containing the name of the closest HTML color to the given color. If there are more than one closest color, print the one which has a smaller associated number in the above table.

## Sample Input

```
120 120 10
111 112 113
5 135 8
-1 -1 -1
```

## Sample Output

```
Olive
Gray
Green
```

# 7014 Ideal Scoreboard

Professor Boffin is a regional contest director of ACM ICPC. He loves watching and analyzing the scoreboard during the contest. He believes that the scoreboard is ideal when all these criteria hold together:

- Each team has solved at least one problem.
- No team has solved all the problems.
- Each problem is solved by at least one team.
- No problem is solved by all the teams.

Obviously, the scoreboard is not ideal at the beginning of the contest, but it may become ideal during the contest. The scoreboard may remain ideal through the end of the contest, or it may lose this property some time later during the contest. In the latter case, it can be shown that it will never become ideal any more. Given the list of the submissions in a regional contest, you must determine the interval in which the scoreboard was ideal.

## Input

The input consists of several test cases. Each test case starts with a line containing 3 space- separated integers $T$, $P$, and $S$ which represent the number of teams, problems, and submissions respectively ($1 \leq T \leq 150$, $1 \leq P \leq 15$, $0 \leq S \leq 5000$). Each of the next $S$ lines represents a contest submission with 4 space-separated parameters:

- $teamID$: the identifier of the team, an integer in the range $[1..T]$.
- $problemID$: the identifier of the problem, an uppercase letter from the first $P$ letters of English alphabet.
- $submission-time$: the time of submission, in '$HH$:$MM$:$SS$' format, all 3 parts are exactly 2 digits ($00 \leq HH \leq 05$, $00 \leq MM, SS \leq 59$).
- result: the result of the submission. It can be one of the following sentences:
  - `Yes`: Only this case shows that the corresponding team has successfully solved the problem.
  - `No - Compilation Error`: Unsuccessful submission due to a compilation error in the submitted program.
  - `No - Wrong Answer`: Unsuccessful submission since the submitted program had a wrong output.
  - `No - Run-time Error`: Unsuccessful submission due to a run-time error during the execution of the submitted program.
  - `No - Time Limit Exceeded`: Unsuccessful submission since the execution of the submitted program did not finish in the time limit.
  - `No - Presentation Error`: Unsuccessful submission due to a formatting error in the output of the submitted program.

No two submissions have the same time. The input terminates with a line containing '`0 0 0`' which should not be processed as a test case.

## Output

For each test case, output a line containing the ideal-interval of the corresponding contest. The interval must be provided with two times in exact '$HH$:$MM$:$SS$' format (as described in the input). The first time shows the moment the scoreboard becomes ideal, and the second time shows the moment the scoreboard is not ideal anymore.

If the scoreboard remains ideal through the end of the contest, the second time must be '`--:--:--`'. If the scoreboard never becomes ideal throughout the contest, both times must be '`--:--:--`'.

## Sample Input

```
2 3 5
1 A 00:10:05 Yes
2 A 00:15:15 No - Wrong Answer
1 C 01:01:01 Yes
2 B 02:20:00 Yes
1 B 03:10:00 Yes
2 3 5
1 A 00:10:05 Yes
2 A 00:15:15 No - Wrong Answer
1 C 01:01:01 Yes
2 B 02:20:00 Yes
1 B 03:10:00 No - Wrong Answer
2 3 5
1 A 00:10:05 Yes
1 C 01:01:01 Yes
2 A 00:15:15 No - Wrong Answer
1 B 03:10:00 Yes
2 B 04:20:00 Yes
0 0 0
```

## Sample Output

```
02:20:00 03:10:00
02:20:00 --:--:--
--:--:-- --:--:--
```

## 7015 Elevators

The new building for the Computer Engineering department has several elevators, but has no stairs. In order to facilitate access to lecture rooms and offices, the manufacturer has set each elevator to stop only at certain pre-defined floors; like some elevators would only stop at odd floors and some only at even floors. But, things are more complicated than this and the buttons inside and outside of each elevator would only work for the set of floors that elevator is assigned to stop at. This has made some improvements in reaching certain destination floors, especially for the faculty members, but has caused a lot of confusions for people like students. If a person $p$ is at floor $i$ and wants to go to floor $j$, which elevator should $p$ take, and to which elevator(s) and on which floor(s) should (s)he transfer to, so that $p$ reaches floor $j$ with minimum travel time? We define $p$'s travel path as $i = f_1 \to f_2 \to \ldots \to f_k = j$, then $p$'s travel time that we want to minimize is

$$\sum_{r=1}^{k-1} |f_i - f_{i+1}|.$$

You have been asked to write an application to help people using these elevators.

### Input

There are multiple test cases in the input. The first line of each test case specifies $n$ $(1 \leq n \leq 10)$, the number of elevators, followed by the source and destination floors. The $i$-th line $(1 \leq i \leq n)$ of the next $n$ lines starts with $m_i$ $(2 \leq m_i \leq 150)$, the number of floors at which the $i$-th elevator may stop, followed by a list of $m_i$ non-negative floor numbers (all numbers are less than 150). The input terminates with a line containing '`0 0 0`' which should not be processed.

### Output

For each test case, output a line containing the minimum travel time that one needs to reach the destination floor when starting from the source floor. It is guaranteed that there is always a possible way from the source floor to the destination floor.

### Sample Input

```
2 2 5
5 0 1 3 5 7
5 0 2 4 6 8
3 3 8
6 0 1 2 3 4 5
5 0 6 7 8 9
4 0 4 5 6
0 0 0
```

### Sample Output

```
7
5
```

# 7016 Antennas

In the 1D-Land which looks like a line, everything is either 1-dimensional or 0-dimensional. Houses and people are just two examples of 1-dimensional or 0-dimensional objects, respectively. Indeed, each house is an interval [a, b] and not surprisingly each person is a point. Due to the running of Maskan Mehr in the 1D-land, every person now owns a house. As a new project, the kind government of the 1D-land plans to provide mobile service so that people can simply do all their jobs remotely from their houses. Due to the increased reputation of the two giant Mobile Telecommunication Companies MTC1 and MTC2, the 1D-land government has decided to donate the new project to MTC1 and MTC2.

As an old strategy, MTC1 and MTC2 first sell their SIM-cards and then start installing their antennas. Based on the services advertised by each company, every person in the 1D-Land has already bought a SIM-card from one of companies. Now each company knows its subscribers and wants to locate its antennas in the 1D-land such that every subscriber's house is covered by at least one antenna.

If a mobile antenna with a covering range of $R$ is located at a point $x$, it covers the interval $[x - R, x + R]$. We say that an antenna covers a house, if the antenna supports the house owner's SIM type and there is at least one point in the house being in the coverage interval of the antenna.

Installing an antenna costs $C_1$ Toomans and $C_2$ Toomans for MTC1 and MTC2, respectively. Since installing antennas is very costly, the two companies have signed an agreement to install **shared antennas** in some positions provided that they reduce the total cost. Installing a **shared antenna** costs $C_3$ Toomans ($\max(C_1, C_2) < C_3 < C_1 + C_2$) and this kind of antenna can support both types of SIM-cards. No matter what the type of antennas is, all antennas have a fixed integer coverage range of $R$.

Given the information of houses and their owners in the 1D-Land and the specification of antennas, your job is to help both companies to position their antennas such that each house is covered by at least one antenna and the total cost gets minimized.

## Input

There are multiple test cases in the input. The first line of each test case contains 5 positive integers $n \leq 5000$ (the number of houses), $R \leq 10^9$ (the range of antennas), $C_1$ (the installing cost of an antenna for MTC1), $C_2$ (the installing cost of an antenna for MTC2) and $C_3$ (the installing cost of a shared antenna). All costs are less than or equal to $10^9$.

The next $n$ lines describe houses; one house per line. A house description is composed of two positive integers $a$ and $b$ ($0 < a \leq b < 10^9$), specifying the house interval $[a, b]$, and an integer with value either 1 and 2 indicating the owner's SIM type where 1 and 2 denote MTC1 and MTC2, respectively. The input terminates with a line containing '`0 0 0 0 0`' which should not be processed.

## Output

For each test case, output a line containing the minimum total cost of installing antennas.

## Sample Input

```
4 10 1000 2000 2400
10 20 1
15 30 2
60 65 1
90 100 2
```

```
0 0 0 0 0
```

## Sample Output

```
5400
```

# 7017 Working Hours

Ali works in a small software company in which the waging method is based on the useful working hours of people (not just their presence). This information is provided by the employees themselves based on a mutual trust. Ali is a rigorous person on such issues. So, during the day, he records all the information needed to calculate the total useful working hours of that day. During months of his work, he has evolved a special format for his recordings. The recording consists of several terms. Each term is a positive or negative time (with separate hours and minutes parts), e.g. `-8:30` or `+09:20`. Ali found that he can express all he needs to record using such terms. For example:

- To show a 90 minute working chunk, he can add either of the terms `+0:90` or `+1:30`.
- If he started a work at 9:00 AM and finished it at 1:30 PM, he can write it with terms `-9:00` and `+13:30`.
- To show a 15 minute break during his job, he can just add a `-0:15` term.
- If he has paused his work at 13:15 and resumed it at 14:05, he can write the break with terms `+13:15` and `-14:05`.

Obviously, the order of the terms does not affect the final result. As a programmer, Ali knows summing up the terms is a boring and error-prone task — something which could be done by computers. Your task is to write a program that does all these summations for Ali. But, entering the recorded terms in the above format was still annoying for Ali. So he tweaked the format a little bit in order to speed up the data entry process:

1. The hour and minute parts are non-negative, but there is no constraint on their maximum value. So, all of the terms `+25:30`, `+24:90`, and `+23:150` are valid and have the same meaning.
2. Typing leading zeros are not necessary. So, `+8:5` is the same as `+08:05`.
3. Typing zero parts are not necessary. So, `+8: = +8:0` and `-:5 = -0:05`.
4. Typing the '`:`' character in each term halves the speed of data entry, because it is the only character which is not available in the number-pad (on the right side of the keyboard) and it also needs the shift key! Thus, from now on, the '`.`' character has the same meaning as the '`:`' character. So, the term `+8.5` means `+8:05`, not `+8:30`!

## Input

There are several test cases in the input. Each test case, consists of several lines (at most 1000 lines) each containing a single term. Test cases are separated with a line containing '`$$$`', and a line containing '`###`' follows the last test case.

## Output

For each test case, output a line containing the sum of the terms. It is guaranteed that the answer is positive. The answer should be written in the '$H$:$MM$' format. So, the minutes part ($MM$) is always a two digit number in range [00..59], and the hours part ($H$) is a non-negative integer with no leading zeros.

## Sample Input

```
-11.45
-:5
-1:10
-.30
-.5
+21.55
$$$
-8.
-1.
-:3
+12.
+.10
###
```

## Sample Output

```
8:20
3:07
```

# 7018 Dragons

There are $N$ cities and $M$ roads in *Dragons Country*. Cities are numbered 1 through $N$, and each road connects exactly two cities. There are totally $K$ dragons $D_1$, $D_2$, ..., $D_K$ in these $N$ cities. Dragon $D_i$ lives in city $C_i$ and has initially $S_i$ heads. It grows $N_i$ new heads every minute while *alive*! A dragon is alive if it has positive number of heads.

We want to hire a number of warriors to kill all dragons. We specify the initial city of each warrior. Then on each minute, each warrior can either go through a road from his current city to an adjacent city, or select an alive dragon in his current city and cut off one of its heads. We can specify the strategy of each warrior on each minute, and after they are done, any alive dragon $D_i$ will grow its $N_i$ new heads.

We want to find the minimum number of warriors with which all dragons can be killed in a finite amount of time.

## Input

There are multiple test cases in the input. For each test case, the first line contains three space-separated integers $N$, $M$ and $K$ ($1 \le N \le 300$, $0 \le M \le N(N-1)/2$, and $1 \le K \le 1000$). Each of the next $M$ lines contains two integers $a$ and $b$ ($1 \le a \ne b \le N$) indicating a road between cities $a$ and $b$. The $i$-th line of the next $K$ lines describes dragon $D_i$ with three space-separated integers $C_i$, $S_i$ and $N_i$ ($1 \le C_i \le N$, $1 \le S_i \le 10^5$, $0 \le N_i \le 10^5$). The input terminates with a line containing '`0 0 0`' which should not be processed.

## Output

For each test case, output a line containing the minimum number of warriors who can eventually kill all dragons.

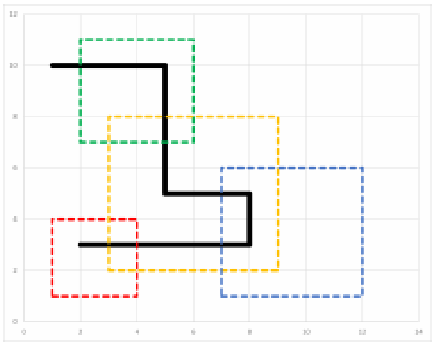
## Sample Input

```
2 1 1
1 2
1 7 4
4 4 2
1 2
2 4
4 1
1 3
1 2 3
2 3 1
0 0 0
```

## Sample Output

```
5
2
```

# 7019 Toll



Due to the increased robbing in the Silk route, the number of merchants travelling through the Silk route is getting less and less. Each robber at any place on the route requests money as much as possible from passing merchants. Now, merchants prefer not to travel through the Silk route even if their travel distances increase by choosing other routes. This has reduced the income of robbers and now Moradbeig, the cheif chair of the robbers, has been forced to set up a new robbing system, namely the toll system. Based on this system, each robber is restricted to a specific square-like area (including the boundary), so-called the robber territory, with no assumption on territories to be disjoint. More importantly, the toll fee is set to be exactly 1 Oshloob inside each territory. The other rules of the new system is listed below.

1. A robber can not get the toll fee outside his territory at all.
2. Each robber must issue a passing ticket, specific to the robber territory, for a merchant who pays the toll fee to him. This ticket allows the merchant to freely travel inside the territory without paying more toll to the other robbers. Once the merchant goes outside the territory, the ticket automatically gets invalid and it can not be used any more.
3. Without any valid ticket, a merchant can not pass through the territories of the robbers.
4. If a merchant likes, he himself can make his current ticket invalid and get a new ticket from any robber whose territory covers him.

Marco Polo, a wealthy merchant, is planning to travel through the Silk route from the beginning to the end. Although the situation is better compared to the past, he still thinks of paying less toll. Your job is to write a program that computes the minimum toll that Marco Polo has to pay in order to traverse the whole route. For simplicity, you can assume the Silk route is a rectilinear path, i.e., each segment of the path is either horizontal or vertical.

## Input

There are multiple test cases in the input. Each test case starts with a line containing two positive integers $n$ and $m$ ($n, m \leq 1000$) which are the numbers of territories and the number of vertices of the Silk route, respectively. The next $n$ lines describe the territories; one territory per line. Each line contains non-negative integers $x$, $y$ and $k$ ($x, y \leq 10^6$, $k \leq 1000$) where $(x, y)$ is the lowest and leftmost corner of the territory and $k$ is the side length of the territory. Each of the next $m$ lines presents the coordinates of the vertices of the Silk route in the order of appearing on the route. It is guaranteed that the route does not intersect itself. The input terminates with a line containing '`0 0`' which should not be processed.

## Output

For each test case, output a line containing the minimum toll that must be paid by Marco Polo.

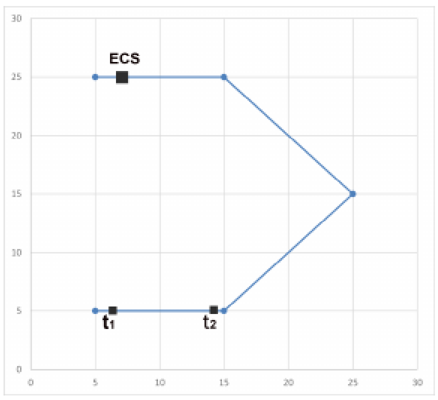
## Sample Input

```
4 6
1 1 3
2 7 4
3 2 6
7 1 5
2 3
8 3
8 5
5 5
5 10
1 10
0 0
```

## Sample Output

```
3
```

# 7020 Robot Race



The International competition of Robot Race 2014 will be held in Tehran. In the competition, a path is specified by the scientific committee, and each robot has to move along the path, from the beginning to the end.

There is an electronic charge station (ECS) on the path, at which robots charge their batteries. Every robot have a device which tells its distance to the ECS. Unfortunately, the devices are not good enough, so, each device shows the Euclidean distance of the robot to the ECS, not the remaining distance on the path to the ECS.

Kamran is a member of the scientific committee of the competition. He knows that there is a common bug in the control software of some robots. A buggy robot imagines that its device shows the remaining distance on the path to the ECS, not the the Euclidean distance to the ECS. As a consequence, from the buggy-robot point of view, its device must show a decreasing sequence of numbers before reaching to the ECS. If this is not the case, the buggy robot crashes since it thinks that it has already passed the ECS without getting charged. Kamran considers a given competition path as unfair, if he can choose a position for ECS on the path such that the buggy robots crash in some time. In other words, a path is unfair if an ECS position can be chosen and there exist three times $t_1 < t_2 < t_3$ such that a robot is at ECS at time $t_3$ and $|p_{t_1}p_{t_3}| < |p_{t_2}p_{t_3}|$ where $|ab|$ denotes the Euclidean distanceb between $a$ and $b$ and $p_t$ is the position of the robot at time $t$. The scientific committee has proposed a list of possible paths for the competition, and Kamran wants to know which path is fair (i.e, the path is not unfair).

## Input

There are multiple test cases in the input. The first line of each test case contains a positive integer $n$ ($n \leq 10,000$), which is the number of points on the path. The next $n$ lines contain $n$ pair of integers $x$ and $y$ ($-10^6 \leq x, y \leq 10^6$). The $i$-th pair specifies the coordinate of the $i$-point in the path. Robots have to start from the first point, and pass through the segments joining consecutive points each after other, and stop when they reach the last point. It is guaranteed that the path does not intersect itself. The input terminates with a line containing '`0`' which should not be processed.

## Output

For each test case, output a line containing either '`Fair`' or '`Unfair`' depending on whether the given path is fair or unfair, respectively.

## Sample Input

```
5
5 5
15 5
25 15
15 25
5 25
```

```
4
0 0
1 0
2 1
3 0
0
```

## Sample Output

```
Unfair
Fair
```

# 7021 Bus

Sharif University of Technology (SUT) has provided a special bus service to take back the interested employees to their homes each day. To use this service, employees should register online. At the beginning of each day, the list of people using the service on that day is posted online making sure that the capacity of the bus is met.

The rent of the bus on each day is p Toomans independent of the number of people in the bus, and as a rule accepted by all, only one of the people in that bus should pay the bus rent to the bus driver. The name of the person responsible to pay the rent is also announced daily.

Your job is to write a program to help SUT select the person responsible for the bus payment in each day such that the assignments become "fair" as defined next.

Assume $L_1$, $L_2$, ..., $L_d$ are the lists of employees in the bus for day 1 to day $d$ and assume that $n_i$ is the size of $L_i$. If a person named $A$ uses the bus for $k$ days $t_1$, $t_2$, ..., and $t_k$, his correct share of the bus rents is $P_A = p*(1/n_{t_1} + 1/n_{t_2} + \ldots + 1/n_{t_k})$ Toomans. If $A$ is selected to pay the bus rent for $r$ times, he will pay $Q_A = r \times p$ Toomans which is $E_A = Q_A - P_A$ Toomans more than his share. The "unfairness" of an assignment is defined to be the maximum of all $E_A$. A "fair" assignment is the one with the minimum unfairness.

## Input

There are multiple test cases in the input. For each test case, the first line contains three positive integers $n$, $d$ and $p$ ($n, d \leq 500, p \leq 10^9$) where $n$, $d$ and $p$ are the number of employees, the number of days and the rent of the bus for each day. The information of each day comes in the next $d$ lines; one line per day. Each line starts with the number of employees using the bus on that day followed by a list of employee IDs which are integer numbers in the range $[1, n]$. For ease of computation, the bus rent $p$ is chosen such that the share of employees in each day is an integer number. The input terminates with a line containing '`0 0 0`' which should not be processed.

## Output

For each test case, output a line containing the unfairness of a fair assignment.

## Sample Input

```
3 2 1000
2 1 2
2 1 3
4 4 3000
2 1 2
2 1 3
2 2 3
3 2 3 4
0 0 0
```

## Sample Output

```
500
2000
```

# 7022 Line Fiting

Fitting a function to a given finite set of points sampled from an unknown function $F : \mathbb{R} \rightarrow \mathbb{R}$ is a basic problem in mathematics. Typical one is to find a linear function $\overline{F}$ that fits the sampled point best. One way to measure how well $\overline{F}$ fits the sample points is defined as follows. Suppose that $F$ is sampled at $x_1$, ..., $x_n$, with $x_1 < \ldots < x_n$. Then the error of $\overline{F}$ is

$$error(F, \overline{F}) = \max_{1 \leq i \leq n} \{|F(x_i) - \overline{F}(x_i)|\}$$

Unfortunately, the function values at the sample points are not known exactly. Instead, we have a discrete probability distribution for each $F(x_i)$, that is, we have a discrete set $y_{i,1}$, ..., $y_{i,m_i}$ of possible values with associated probabilities $p_{i,j}$ such that $\mathbf{Pr}[F(x_i) = y_{i,j}] = p_{i,j}$. We define the error measure in the following natural way using the concept of the expected value:

$$error(F, \overline{F}) = \max_{1 \leq i \leq n} \{\mathbf{E}[|F(x_i) - \overline{F}(x_i)|]\}$$

The goal is now to find a linear function $\overline{F} = ax + b$ that minimizes the error. You must write a program that gets the sample points and their probabilities and computes the minimum error defined above.

## Input

There are multiple test cases in the input. The first line of each test case contains $n$, the number of the sampled points ($1 \leq n \leq 10^5$). Next, the information of sampled points (in the increasing order) comes in $n$ lines; one line for each sampled point. For the $i$-th sampled point, the line starts with two non-negative integer numbers $x_i$ and $m_i$ where $x_i$ is the value at which we sample the function and $m_i$ is the size of distribution ($1 \leq m_i \leq 10$ and $0 \leq x_i \leq 10^9$). Then it is followed by a list of $m_i$ non-negative numbers being less than $10^9$ which are the function values at $x_i$, and finally a list of $m_i$ probabilities. For simplicity, each given probability $p$ in the input is a non-negative integer less than or equal to 100. You can get the real probability by dividing $p$ to 100. You can assume the summation of all probability numbers is equal to 100. The input terminates with a line containing '0' which should not be processed.

## Output

For each test case, output a line containing the minimum error rounded to exactly one digit after the decimal point.

## Sample Input

```
2
0 2 0 1 50 50
1 2 0 1 50 50
0
```

## Sample Output

```
0.5
```