Problem A
Being Late

Professor AlwaysLate is notorious for having something to say past the end of class time, and even past the start time of following classes. This habit is not viewed kindly by the students' rights movement, which motivated them to collect data about such conduct. The students' rights movement wants to process their extensive data collection and extract some useful information for their annual report, and you are asked to help. Your task is to write a program to read a number of lines, where each line contains a record about a single class, and calculate the average length of time spent past the end of class for Professor AlwaysLate.

INPUT Format:

The input starts with a positive integer N that represents the number of records, with $30000 \geq N > 0$, on a separate line followed by a description of the N records. Each record is described on a separate line by: Four (4) integers oHour, oMinute, aHour and aMinute. The integers are separated by single spaces. oHour and oMinute represent the official finish time of a class, while aHour and aMinute represent the actual time Professor AlwaysLate left the classroom, where $0 \leq$ aMinute, oMinute < 60, $8 \leq$ oHour ≤ 20 and oHour $\leq$ aHour $\leq$ oHour +1.

OUTPUT Format:

The output of your program is a single integer that represents the average number of minutes that Professor AlwaysLate spends lecturing past the end of his official class finish time rounded down (that is, truncated) to the nearest minute.

Input-Output Examples:

Example input A1:	Example output A1:
4 10 50 10 55 12 50 13 0 17 50 17 50 13 45 14 0	7

It is worth mentioning that the students' rights movement considers early finishing of a class as a class finished on time, as demonstrated in the following example:

Example input A2:	Example output A2:
4 10 50 10 55 12 50 13 0 17 50 17 40 13 45 14 0	7

Problem B
Money Lesson

Mrs Brown is teaching her pupils about money. At the moment, they are using dummy $5, $10 and $20 notes. Before a lesson starts, Mrs Brown arranges piles of different notes for the pupils so that each one has the same amount of money. The total number of notes in each pile does not exceed 100 notes. Then she takes one, and only one, note from one pupil's pile and puts it in the pile of another pupil. The pupils then have to work out which one has the most money, and which one has the least. Sometimes, Mrs Brown does not move a note so that each pupil has exactly the same amount of money – the pupils have to be able to tell when this happens. Your task is to write a program to calculate the correct answer that Mrs Brown's pupils ought to report.

INPUT Format:

The input consists of a series of scenarios for a number of lessons. The first line in each scenario consists of a positive integer, N, which represents the number of pupils in the class that day ($2 < N < 30000$). Each of the following N lines contains the data for one pupil. The lines contain four items, the name of the pupil (a single series of between 2 and 10 letters, lower case except for the first) followed by the number of $5, $10 and $20 notes (in that order) allocated to that pupil. Items are separated by single spaces. Input is terminated by a scenario where N equals -1. This scenario should not be processed.

OUTPUT Format:

Output consists of one line for each scenario. It will be in one of the following two formats:

X has most, Y has least money.
All have the same amount.

X and Y are student names.

Input-Output Examples:

Example Input B:	Example Output B:
5 Andrew 0 0 2 Brenda 0 4 0 Chen 8 1 0 David 2 0 1 Eloise 0 2 1 3 Xerxes 0 3 0 Yolanda 0 1 1 Zebedee 4 1 0 -1	Chen has most, David has least money. All have the same amount.

Problem C
Explorer's Diary

Marico, an eccentric explorer of the orient, wrote an extensive diary of his trips. He did not mean to encrypt them, but that is what people thought due to his unusual writing style. Marico's writings proceeded from left to right but wrote each word vertically, and used the 26 letters of the alphabet (upper case) only. For example, he wrote the following:

```
THIS IS MY FIRST TRIP TO THE ORIENT I FOUND
THEIR WRITING SO BEAUTIFUL THAT I ADOPTED INTO
MY OWN
```

as

T	I	M	F	T	T	T
H	S	Y	I	R	O	H
I			R	I		E
S	I	F	S	P	S	
		O	T		O	B
O	I	U		W		E
R		N	T	R	O	A
I		D	H	I	W	U
E			E	T	N	T
N		A	I	I		I
T		D	R	N		F
		O		G		U
T		P	I			L
H		T	N	M		
A		E	T	Y		
T		D	O			

Marico's diary has been digitized for the sake of preserving it, but its content remained a mystery until his unique writing style was understood. The process of converting a large number of pages back was found to be tedious and risky for error-prone human volunteers, and the decision was taken to develop a software tool to perform the task instead.

Your task is to write a program for the purpose of converting pages of Marico's diary into the familiar way of writing such that:

1. words on the same line are separated by *exactly* one blank space,
2. width of the output text does not exceed a specified value,
3. words are not to be broken between output lines.

For example, your program should convert the page of Marico's diary shown on the left to the text shown on the right when the width is bounded by 10 places.

```
TIMFTTT
HSYIROH
I  RI E
SIFSPS
  OT OB
OIU W E
R NTROA
I DHIWU
E  ETNT
N AII I
T DRN F
  O G U
T PI  L
H TNM
A ETY
T DO
```

converts into

```
THIS IS MY
FIRST TRIP
TO THE
ORIENT I
FOUND
THEIR
WRITING SO
BEAUTIFUL
THAT I
ADOPTED
INTO MY
OWN
```

INPUT Format:

The first line of the input consists of the two positive integers L and W, separated by a single space. L is the maximum possible number of lines on each page of Marico's diary, and W is the desired width of the output text. $40 \geq L > 0$ and $40 \geq W > 0$. The rest of the input consists of a series of one or more page descriptions.

The first line in each page description consists of a positive integer, N, $1 \leq N \leq 20$, which represents the number of columns. Each of the following lines, if any, contains exactly N characters. None of the pages completely consists of blank characters, and none of the pages end with a blank line; that is, the last line contains at least one character that is not blank. The page description is terminated with a line that contains only a single '#' character, and should not be formatted.

The input is terminated with a line that contains only the string "##" (that is, two '#' characters), and should not be processed.

OUTPUT Format:

For each page the output consists of number of lines. The first line contains the page number starting with the value of one (1), as shown in the "Example output C" below, and then followed by

1. the lines of the formatted text, or
2. the sentence "`This page is empty.`" on the next line.

Input-Output Examples:

Example Input C:	Example Output C:

40 10
7
TIMFTTT
HSYIROH
I RI E
SIFSPS
  OT OB
OIU W E
R NTROA
I DHIWU
E ETNT
N AII I
T DRN F
  O G U
T PI L
H TNM
A ETY
T DO
#
8
#
2
AL
FU
TN
EC
RH
 	
WL
EE
 F
 T
#
##
	Page number 1
	THIS IS MY
	FIRST TRIP
	TO THE
	ORIENT I
	FOUND
	THEIR
	WRITING SO
	BEAUTIFUL
	THAT I
	ADOPTED
	INTO MY
	OWN
	Page number 2
	This page is empty.
	Page number 3
	AFTER
	LUNCH WE
	LEFT

Problem D
Pattern Finder

A number pattern is a sequence of two or more numbers that satisfies a certain rule. For instance, the sequence "3, 5, 7, 9, …" is a number pattern that starts with three and jumps by twos. In primary school, number patterns are used as a tool for learning basic arithmetic and problem-solving skills. Finding a number pattern in a grid of integers is Mrs Brown's favourite challenge for her pupils.

Mrs Brown's number pattern challenge for her 2nd grade pupils is to find a number pattern in a grid of integers. The number pattern must start at the smallest value, jumps to an adjacent cell in the grid by adding a fixed positive value, and terminates at the largest value. Two cells are considered adjacent if they share the same row or share the same column.

5	4	6	9
10	7	8	4
13	9	10	12
12	14	12	14

8	4	6	9
5	7	8	4
10	9	10	12
15	14	12	14

For instance, the grid on the left has a pattern that starts at 4 and jumps by 2 and terminates at the largest value of 14, but no pattern can be found in the grid on the right.

Mrs Brown has a large number of these grids, but she does not have the time to find the correct answer for each one. Your task is to help by writing a program to check the existence of a pattern of the required form in a given grid.

INPUT Format:

The input consists of a series of scenarios. The first line in each scenario consists of two positive integers N and M that represent the number of rows and columns in a grid. $1 \leq N, M \leq 100$. Each of the following N lines contains M integers that represent the integers in a single row in the grid. The integers are separated by single spaces, and have values that does not exceed 30000.

Input is terminated by a challenge where N and M equal -1. This challenge should not be processed.

OUTPUT Format:

Output consists of one line for each grid. It will be in one of the following two formats:

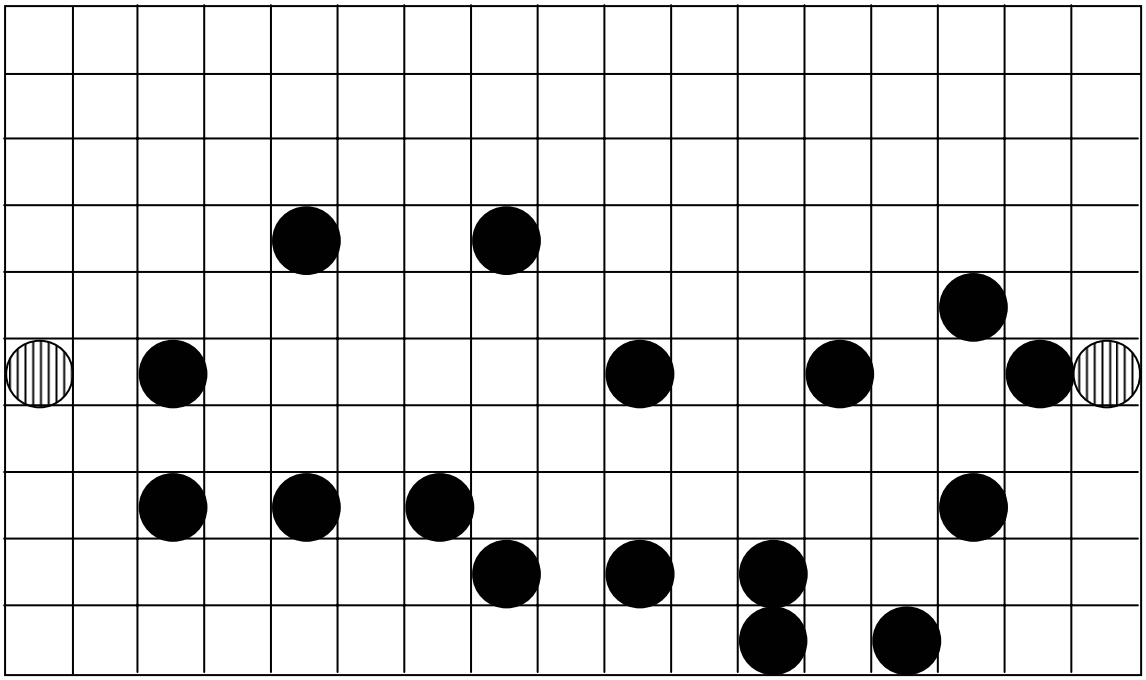
At least one pattern exists.
No pattern can be found.

Input-Output Examples:

Example input D:	Example output D:
2 8 77 78 79 77 77 77 77 77 78 79 80 81 82 83 84 85 -1 -1	At least one pattern exists.

Problem E
Takeshi Castle

Takeshi Castle was a Japanese game show that aired from 1986 to 1989 on the Tokyo Broadcasting System. It featured the esteemed Japanese actor Takeshi Kitano as a count who owns a castle and sets up impossible challenges for a volunteer army to get to him. The show has become a television hit around the world. An old fan of the show asked us for help with the design of a challenge, called **Lily Pads**[‡], to use as a game following the upcoming ICPC Finals. As a challenge cannot be suitable for a Takeshi Castle game unless it is biased against the volunteer army, your task is to write a program to *verify* that a given placement of the discs will cause the majority of the volunteer army members to fail the challenge. The maximum distance that each member of the army can leap, to jump from one disc to another, has been collected by Takeshi spies and will be made available to us. The largest such distance does not exceed the value of 10.



[‡] The **Lily Pads** challenge is to cross a body of cold water by jumping between circular discs, with legs tied together, without falling into the water.

For example: the above setting with each grid cell being of size 1x1, is a suitable challenge against a volunteer army of five members with three members capable of jumping a maximum distance of two units and two members capable of jumping a maximum of three units as the majority of volunteers will fail the challenge. Note that the distance between two discs is defined to be the length of the line connecting their centers. In the plane, the distance between points (x_1,y_1) and (x_2,y_2) is given by

$$((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}$$

INPUT Format:

The input consists of a series of challenges. The first line in each challenge consists of the positive integers N and M, separated by one space. N is the number of the volunteers and M is the number of discs in the lily pad pond, where $30000 \geq N > 0$ and $50 \geq M \geq 2$. The second line consists of N positive integers that represent the maximum distances that the N members of the volunteer army can leap. The third line consists of 4 positive integers that represent the *x*- and *y*-coordinates of the *start* and *finish* discs[§]. Each of the following M-2 lines contains two positive integers that represent the *x*- and *y*-coordinates of the remaining discs. The integers are separated by single spaces. The values of all *x*- and *y*-coordinates do not exceed 500.

Input is terminated by a challenge where N and M equal -1. This challenge should not be processed.

OUTPUT Format:

Output consists of one line for each challenge. It will be in one of the following two formats:

NOT valid
valid

[§] The *start* and *finish* discs are shown striped in the example.

Input-Output Examples:

Example Input E:	Example Output E:
5 18 3 3 3 3 3 1 5 17 5 3 5 3 3 10 5 13 5 16 5 5 3 5 7 7 3 8 2 8 7 10 2 12 1 12 2 14 1 15 3 15 6 6 18 1 2 1 3 1 2 1 5 17 5 3 5 3 3 10 5 13 5 16 5 5 3 5 7 7 3 8 2 8 7 10 2 12 1 12 2 14 1 15 3 15 6 -1 -1	NOT valid valid

Problem F
Balanced Races

The organizing committee of a popular charity fair in a small town wants to include walking races that can be run in a competitive format. The idea is that competitiveness will make the races more entertaining, attract more viewers and thus more contributions. The proposed new format is to divide the walkers into heats that are competitive; that is, the time difference between the first and last participants to arrive at finish line in each heat is as small as possible. The heats are then scheduled to run independently through the day of the fair. The organizers will provide us with information about the number of registered walkers along with the best available estimate of the times they take to finish the race course, the number of heats they can schedule and the maximum number of participants allowed in each heat. Your task is to read the provided information and then organize the participants into heats that satisfy the given constraints and minimizes over all the heats the maximum time difference within a heat, or warn the organizers that their constraints cannot be satisfied.

INPUT Format:

The input consists of a series of races. The first line in each race description consists of three positive integers N, M and Q that are separated by single spaces. N represents the largest number of walkers allowed to participate in each heat, M represents the largest number of heats that the organizers can schedule and Q is the number of walkers registered in the race. Each of the following Q lines contains an integer T, $0 < T \leq 10000$, that represents the known time for one of the registered walkers to finish the race course. $1 \leq N, M \leq 50$ and $1 \leq Q \leq 1000$.

Input is terminated by a race with N, M and Q equal to -1. This race should not be processed.

OUTPUT Format:

For each race the output consists of a single line that contains the race number starting with the value of one (1), followed by a ": ", as shown in the "Example output F" below, and then followed by one of the following formats:

1. N
2. Organization Fault!

N is the minimum, over all the heats, of maximum difference of finish times within each heat.

Input-Output Examples:

Example input F:	Example output F:
3 2 3 21 20 13 3 3 12 21 18 10 16 10 10 89 11 11 6 8 3 -1 -1 -1	Race 1: 1 Race 2: Organization Fault!

Problem G
Smoker's Walk

Ms Millerose is an old hand in the local newspaper business. Throughout her whole career, she has worked in a big hall where each reporter occupies one of the desks neatly organized into a grid. The typewriters are now gone and have been replaced by computers, but that does not bother her. The other change, a non-smoking work environment, has forced her to satisfy her need to smoke, outside of the building regularly. However the stench of smoke that follows her on the walk back to her desk has been a source of inconvenience to her co-workers. Ms Millerose would like to learn how to plan her walk from the hall's entrance, which is located at the north-west corner of the hall, to her desk, so as to minimize the inconvenience to the reporters working at their desks at the time. Your task is to help Ms Millerose, and others in similar situations, by writing a program to plan a smoker's walk in a generic setting that may be described as follows:

> Given a grid of N rows and M columns with a subset Q of the grid points occupied, Q < N*M. Write a program to find a path from the grid point (1,1), which is located in the upper-left point in the grid, to another grid point (*r*,*c*) such that:
>
> 1. the path consists of a sequence of grid points with the constraint that consecutive points share a row or share a column,
> 2. the path does not contain a grid point in the subset Q,
> 3. the sum of distances between the points in Q and the path is maximized.

All office descriptions have been designed such that at least one path exists between (1,1) and (*r*,*c*).

The destination grid point is identified by its row number *r* and its column number *c*. The distance between a grid point **X** and a path **P** in the grid is the minimum number of grid edges required to travel between **X** and **P**.

INPUT Format:

The first line of the input contains a positive integer that represents the number of generic office descriptions that follow. The first line in each office description consists of two positive integers N and M that represent the number of rows and columns in the grid. The integers are separated by a single space. The second line contains a positive integer Q that represents the number of currently occupied grid points. Each of the following Q lines contains two integers that represent the row number *r* and column number *c* of an occupied point. The last line contains two integers that represent the row number *r* and column number *c* of the smoker's destination point. The integers are separated by single spaces. $1 < N \leq 30$, $1 < M \leq 30$, and $1 \leq Q \leq 100$.

OUTPUT Format:

For each generic office description the output consists of a single line that contains the number starting with the value of one (1), followed by a ": ", as shown in the "Example output G" below, and then followed by an integer N that is the measure of inconvenience a smoker's walk will cause to her office mates.

Input-Output Examples:

Example input G:	Example output G:
4 10 10 4 5 5 3 5 4 6 2 2 8 8 10 10 4 5 5 4 4 3 3 2 2 8 8 10 10 4 1 5 1 4 1 3 1 2 8 8 10 10 4 5 5 4 9 3 3 2 10 8 8	Office 1: 14 Office 2: 10 Office 3: 10 Office 4: 19

Problem H
P2P Currency Service

The *Exotica* travel agency specializes in organizing foreign holiday trips. *Exotica* want to provide a new service of supplying cash to its customers, before the start of their holidays, in the currency of their destination. Towards that goal, the agency introduced a new P2P (peer-to-peer) currency trading model that allows it to provide this service without taking financial risks or maintaining an inventory of exotic currencies.

The proposed P2P model is based on creating an Internet site for its customers to send their currency trading requests. *Exotica* task is limited to the identification of pairing compatible requests, and then putting the two customers in touch to negotiate a mutually agreeable exchange rate. Your task is to write a program that identifies the largest number of pairings among the received requests. *Exotica* want to discourage commercial trading operators from infiltrating the service, and require that the system restricts each customer to a single trade.

You are being asked to write a program that reads the currency trading requests, then calculates the largest number of possible pairings between them with the constraints that

1. each customer is restricted to a single trade, and
2. no customer is allowed to trade with himself or herself (as shown in the "Example output H" below)

INPUT Format:

The first line of the input contains a positive integer that represents the number of trading rounds that follow. Each round description consists of "1+*n*" lines. The first of the "1+*n*" lines contains one integer, *n*, which represents the number of currency exchange requests to be processed, where $2 \leq n \leq 300$. Each currency exchange request, which is given in a separate line, contains the name of a customer given as a character string of length between 2 and 10, inclusive, characters without any white (space, tab …) characters, the names of two currencies given as strings. The first currency string represents the available currency and the second currency string represents the required currency, with each string containing no more than 5 characters without any white characters. The strings are separated by single spaces.

OUTPUT Format:

For each round of trade, the output is a single line that contains the maximum number of possible customer pairings given as an integer.

Input-Output Examples:

Example input H:	Example output H:
2 7 Alice USD EUR Bruce CNY USD Bruce CNY EUR MingLi CNY EUR Bruce EUR USD MingLi EUR CNY Li EUR CNY 2 Bruce EUR USD Bruce USD EUR	2 0

Problem I
Currency Shopping

After traveling widely over many decades, *Bint Fatuma* has accumulated a fortune in many different currencies. Now she wants to settle down, to convert her fortune into the local currency and use the money to start a business. It will be beneficial to get the maximum possible amount out of this conversion. After taking little more than a casual look at the exchange rates between different currencies, *Bint Fatuma* suspects that the most advantageous way for converting one currency into another may not always be the direct way. For instance, if she starts with a Euro, change it into dollars and then converts the dollars into pounds, she may end up with more pounds than by converting a Euro directly into pounds.

Given the set of currencies be c_1, c_2, c_3,…., c_n, available for trading with $r_{i,j}$ being the exchange rate between the two currencies c_i and c_j,; that is, she can purchase $r_{i,j}$ units of currency c_j in exchange for one unit of currency c_i. You have been assigned the task of helping *Bint Famuta* by writing a program that finds the most advantageous sequence of currency exchanges for converting one currency into another for a given set of exchange rates. Better yet, if your program can find a sequence of currency exchanges with the property that $r_{a,b} * r_{b,d} *….* r_{h,a} > 1$, then she can make a profit by continuously performing this sequence of currency conversions until the bank goes broke!

INPUT Format:

Input for this problem starts with a positive integer K that represents the number of scenarios on a separate line followed by a description of the K scenarios. $0 < K \leq 20$.

The first line in each scenario consists of a positive integer N that represents the number of the tradable currencies, with $2 \leq N \leq 25$.

The second line consists of N strings that represent the names of the tradable currencies. Each string consists of 3 characters. The strings are separated by single spaces. This is followed by N lines, each containing the exchange rates from one currency. These lines are in the order of the currencies' appearance in the second line described above. In each line the "to" currencies are also in that order. For instance, in the scenarios in the example input, each second line of exchange rates contains the exchange rates from USD into the FEP, then to itself (which naturally has the value of 1.0000), then to the CAD and finally to the AUS. Each of the N line consists of N floating-point numbers that represent currency exchange rates. The value of each number has a positive value that does not exceed the value of five and contains four decimal places. The numbers are separated by single spaces.

The last line in each scenario contains two strings. Each string consists of 3 characters and represents a currency. The strings are separated by a single space.

OUTPUT Format:

The output consists of one line for each scenario. It will be in one of the following two formats:

1. a number that represents the best exchange rate from the first currency given in the last line of input in this scenario to the second currency given in the last line of input in this scenario. The number must be rounded down (that is, truncated) to the nearest four decimal places, or
2. The string "Break the bank!"

Input-Output Examples:

Example input I:	Example output I:
3 4 FEP USD CAD AUS 1.0000 3.7209 3.9070 4.3053 0.2667 1.0000 1.0418 1.1480 0.2536 0.9500 1.0000 1.1019 0.2300 0.8700 0.9050 1.0000 FEP AUS 4 FEP USD CAD AUS 1.0000 3.7209 3.9070 4.1053 0.2667 1.0000 1.0418 1.1480 0.2536 0.9500 1.0000 1.1019 0.2300 0.8700 0.9050 1.0000 FEP AUS 4 FEP USD CAD AUS 1.0000 3.7209 3.9070 4.3053 0.3667 1.0000 1.0418 1.1480 0.2536 0.9500 1.0000 1.1019 0.2300 0.8700 0.9050 1.0000 FEP AUS	4.3053 4.3051 Break the bank!