10140 - Prime Distance

Time limit: 3.000 seconds
The branch of mathematics called number theory is about properties of numbers. One of the areas that has captured the interest of number theoreticians for thousands of years is the question of primality. A prime number is a number that is has no proper factors (it is only evenly divisible by 1 and itself). The first prime numbers are 2,3,5,7 but they quickly become less frequent. One of the interesting questions is how dense they are in various ranges. Adjacent primes are two numbers that are both primes, but there are no other prime numbers between the adjacent primes. For example, 2,3 are the only adjacent primes that are also adjacent numbers.

Your program is given 2 numbers: L and U (1<=L<U<=2,147,483,647), and you are to find the two adjacent primes C1 and C2 (L<=C1<C2<=U) that are closest (i.e. C2-C1 is the minimum). If there are other pairs that are the same distance apart, use the first pair. You are also to find the two adjacent primes D1 and D2 (L<=D1<D2<=U) where D1 and D2 are as distant from each other as possible (again choosing the first pair if there is a tie).

Input

Each line of input will contain two positive integers, L and U, with L < U. The difference between L and U will not exceed 1,000,000.

Output

For each L and U, the output will either be the statement that there are no adjacent primes (because there are less than two primes between the two given numbers) or a line giving the two pairs of adjacent primes.

Sample Input

```
2 17
14 17
```

Output for Sample Input

```
2,3 are closest, 7,11 are most distant.
There are no adjacent primes.
```

10219 - Find the ways !

Time limit: 3.000 seconds

An American, a Frenchman and an Englishwoman had been to Dhaka, the capital of Bangladesh. They went sight-seeing in a taxi. The three tourists were talking about the sites in the city. The American was very proud of tall buildings in New York. He boasted to his friends, "Do you know that the Empire State Building was built in three months?"

"Really?" replied the Frenchman. "The Eiffel Tower in Paris was built in only one month! (However, The truth is, the construction of the Tower began in January 1887. Forty Engineers and designers under Eiffel's direction worked for two years. The tower was completed in March 1889.)

"How interesting!" said the Englishwoman. "Buckingham Palace in London was built in only two weeks!!"

At that moment the taxi passed a big slum (However, in Bangladesh we call it "Bostii"). "What was that? When it was built ?" The Englishwomen asked the driver who was a Bangladeshi.

"I don't know!" , answered the driver. "It wasn't there yesterday!"

However in Bangladesh, illegal establishment of slums is a big time problem. Government is trying to destroy these slums and remove the peoples living there to a far place, formally in a planned village outside the city. But they can't find any ways, how to destroy all these slums!

Now, can you imagine yourself as a slum destroyer? In how many ways you can destroy ***k*** slums out of ***n*** slums ! Suppose there are 10 slums and you are given the permission of destroying 5 slums, surly you can do it in 252 ways, which is only a 3 digit number, Your task is to find out the digits in ways you can destroy the slums !

The Input. The input file will contain one or more test cases.

Each test case consists of one line containing two integers n (n>=1) and k (1<=<k=<n).

The Output. For each test case, print one line containing the required number. This number will always fit into an integer, i.e. it will be less than 2^31-1.

Sample Input

```
20 5
100 10
200 15
```

Sample Output

```
5
14
23
```

10199 - Tourist Guide

Time limit: 3.000 seconds

Rio de Janeiro is a very beautiful city. But there are so many places to visit that sometimes you feel a bit lost. But don't worry, because your friend Bruno promised you to be your tourist guide. The problem is that he is not a very good driver, as he can't see very well (poor Bruno).

Because of his disabilities, Bruno have a lot of fines to pay and he doesn't want to have even more to pay, even though he promised you to be your tourist guide. What he would like to know, however, is where are all the cameras that help the police to fine the bad drivers, so he can drive more carefully when passing by them.

Those cameras are strategically distributed over the city, in locations that a driver must pass through in order to go from one zone of the city to another. In order words, if there are two city locations A and B such that to go from one to another (A to B or B to A) a driver must pass through a location C, then C will have a camera.

For instance, suppose that we have 6 locations (A, B, C, D, E and F) and that we have 7 routes (all of them bidirectonal) B-C, A-B, C-A, D-C, D-E, E-F and F-C. There's a camera on C because to go from A to E, for instance, you must pass through C. In this configuration, there's only one camera (on C).

Your task is to help Bruno (as he wants to be a musician and he doesn't want to get even close of computers) and write a program which will tell him where are all the cameras, given the map of the city, so he can be your tourist guide and avoid further fines.

The Input

The input will consist on an arbitrary number of city maps. Each city map will begin with an integer N (2 < N <= 100) which stands for the total locations of the city. Then will follow N different place names, one per line. Each place name will have at least one and at most 30 characters (all of them will be lowercase latin letters). Then will follow a non-negative integer R which stands for the total routes of the city. Then will follow R lines with the routes. Each route will be represented by the name of both places that the route connects (remember that the routes are bidirectional). Location names in route description will always be valid and there will be no route from one place to itself. You must read until N = 0, and this input should not be processed.

The Output

For each city map you must print the line:

```
City map #d: c camera(s) found
```

Where d stands for the city map number (starting from 1) and c stands for the total number of cameras. Then should follow c lines with the location names where are each camera (in alphabetic order). You should print a blank line between output sets.

Sample Input

```
6
sugarloaf
maracana
copacabana
ipanema
corcovado
lapa
7
ipanema copacabana
copacabana sugarloaf
ipanema sugarloaf
maracana lapa
sugarloaf maracana
corcovado sugarloaf
lapa corcovado
5
guanabarabay
downtown
botanicgarden
colombo
sambodromo
4
guanabarabay sambodromo
downtown sambodromo
sambodromo botanicgarden
colombo sambodromo
0
```

Sample Output

```
City map #1: 1 camera(s) found
sugarloaf

City map #2: 1 camera(s) found
sambodromo
```

10842 - Traffic Flow

Time limit: 3.000 seconds

"Arnie Pie in the sky with the morning commute. Traffic this morning is as bad as it gets. Due to a fire at the army testing lab a bunch of escaped, infected mokeys are roaming the expressway. Despite the swealtering heat, don't unroll your windows 'cause those monkeys seem confused and irritable."

Arnie Pie, "The Simpsons."

A city has **n** intersections and **m** bidirectional roads connecting pairs of intersections. Each road has a certain traffic flow capacity, measured in cars per minute. There is a path from every intersection to every other intersection along some sequence of roads. The road maintenance department is over budget and needs to close as many roads as possible without disconnecting any intersections. They want to do it in such a way that the minimum capacity among all of the remaining roads is as large as possible.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (0<**n**<=100) and **m** (0<**m**<=10000). The next **m** lines will describe the m roads, each one using 3 integers, **u**, **v** and **c** (0<=**u**,**v**<**n**), (0<**c**<=1000). **u** and **v** are the endpoints of the road and **c** is its capacity.

Output

For each test case, output one line containing "Case #**x**:" followed by the capacity of the minimum-capacity remaining road.

Sample Input	Sample Output
2 2 3 0 1 10 0 1 20 0 0 30 4 5 0 1 1 3 1 2 1 2 3 2 3 4 0 2 5	Case #1: 20 Case #2: 3

334 - Identifying Concurrent Events

Time limit: 3.000 seconds

It is important in distributed computer systems to identify those events (at identifiable points in time) that are concurrent, or not related to each other in time. A group of concurrent events may sometimes attempt to simultaneously use the same resource, and this could cause problems.

Events that are not concurrent can be ordered in time. For example, if event e_1 can be shown to always precede event e_2 in time, then e_1 and e_2 are obviously not concurrent. Notationally we indicate that e_1 precedes e_2 by writing $e_1 \rightarrow e_2$. Note that the precedes relation is transitive, as expected. Thus if $e_1 \rightarrow e_2$ and $e_2 \rightarrow e_3$, then we can also note that $e_1 \rightarrow e_3$.

Sequential events in a single computation are not concurrent. For example, if a particular computation performs the operations identified by events e_1, e_2 and e_3 in that order, then clearly $e_1 \rightarrow e_2$ and $e_2 \rightarrow e_3$.

Computations in a distributed system communicate by sending messages. If event *esend* corresponds to the sending of a message by one computation, and event *erecv* corresponds to the reception of that message by a different computation, then we can always note that $esend \rightarrow erecv$, since a message cannot be received before it is sent.

In this problem you will be supplied with lists of sequential events for an arbitrary number of computations, and the identification of an arbitrary number of messages sent between these computations. Your task is to identify those pairs of events that are concurrent.

Input

A number of test cases will be supplied. For each test case the input will include first an integer, *NC*, specifying the number of computations in the test case. For each of these *NC* computations there will be a single line containing an integer NE_i that specifies the number of sequential events in the computation followed by NE_i event names. Event names will always contain one to five alphanumeric characters, and will be separated from each other by at least one blank. Following the specification of the events in the last computation there will be a line with a single integer, *NM*, that specifies the number of messages that are sent between computations. Finally, on each of the following *NM* lines there will be a pair of event names specifying the name of the event associated with the sending of a message, and the event associated with the reception of the message. These names will have previously appeared in the lists of events associated with computations, and will be separated by at least one blank. The last test case will be followed by the single integer 0 on a line by itself.

Output

For each test case, print the test case number (they are numbered sequentially starting with 1), the number of pairs of concurrent events for the test case, and any two pair of the concurrent event names. If there is only one concurrent pair of events, just print it. And if there are no concurrent events, then state that fact.

Example

Consider the following input data:

```
2
2 e1 e2
2 e3 e4
1
e3 e1
0
```

There are two computations. In the first `e1` $\rightarrow$ `e2` and in the second `e3` $\rightarrow$ `e4`. A single message is sent from `e3` to `e1`, which means `e3` $\rightarrow$ `e1`. Using the transitive property of the precedes relation we can additionally determine that `e3` $\rightarrow$ `e2`. This leaves the pairs (`e1`,`e4`) and (`e2`,`e4`) as concurrent events.

Sample Input

```
2
2 e1 e2
2 e3 e4
1
e3 e1
3
3 one two three
2 four five
3 six seven eight
2
one four
five six
1
3 x y zee
0
2
2 alpha beta
1 gamma
1
gamma beta
0
```

Sample Output

```
Case 1, 2 concurrent events:
(e1,e4) (e2,e4)
Case 2, 10 concurrent events:
(two,four) (two,five)
Case 3, no concurrent events.
Case 4, 1 concurrent events:
(alpha,gamma)
```

11369 - Shopaholic

Time limit: 1.000 seconds

Lindsay is a shopaholic. Whenever there is a discount of the kind where you can buy three items and only pay for two, she goes completely mad and feels a need to buy all items in the store. You have given up on curing her for this disease, but try to limit its effect on her wallet.

You have realized that the stores coming with these offers are quite selective when it comes to which items you get for free; it is always the cheapest ones. As an example, when your friend comes to the counter with seven items, costing 400, 350, 300, 250, 200, 150, and 100 dollars, she will have to pay 1500 dollars. In this case she got a discount of 250 dollars. You realize that if she goes to the counter three times, she might get a bigger discount. E.g. if she goes with the items that costs 400, 300 and 250, she will get a discount of 250 the first round. The next round she brings the item that costs 150 giving no extra discount, but the third round she takes the last items that costs 350, 200 and 100 giving a discount of an additional 100 dollars, adding up to a total discount of 350.

Your job is to find the maximum discount Lindsay can get.

Input

The first line of input gives the number of test scenarios, $1 \leq t \leq 20$. Each scenario consists of two lines of input. The first gives the number of items Lindsay is buying, $1 \leq n \leq 20000$. The next line gives the prices of these items, $1 \leq p_i \leq 20000$.

Output

For each scenario, output one line giving the maximum discount Lindsay can get by selectively choosing which items she brings to the counter at the same time.

Sample Input

```
1
6
400 100 200 350 300 250
```

Sample Output

```
400
```

10369 - Arctic Network

Time limit: 3.000 seconds

Problem C: Arctic Network

The Department of National Defence (DND) wishes to connect several northern outposts by a wireless network. Two different communication technologies are to be used in establishing the network: every outpost will have a radio transceiver and some outposts will in addition have a satellite channel.

Any two outposts with a satellite channel can communicate via the satellite, regardless of their location. Otherwise, two outposts can communicate by radio only if the distance between them does not exceed D, which depends of the power of the transceivers. Higher power yields higher D but costs more. Due to purchasing and maintenance considerations, the transceivers at the outposts must be identical; that is, the value of D is the same for every pair of outposts.

Your job is to determine the minimum D required for the transceivers. There must be at least one communication path (direct or indirect) between every pair of outposts.

The first line of input contains N, the number of test cases. The first line of each test case contains 1 <= S <= 100, the number of satellite channels, and S < P <= 500, the number of outposts. P lines follow, giving the (x,y) coordinates of each outpost in km (coordinates are integers between 0 and 10,000). For each case, output should consist of a single line giving the minimum D required to connect the network. Output should be specified to 2 decimal points.

Sample Input

```
1
2 4
0 100
0 300
0 600
150 750
```

Sample Output

```
212.13
```

10130 - SuperSale

Time limit: 3.000 seconds

There is a SuperSale in a SuperHiperMarket. Every person can take only one object of each kind, i.e. one TV, one carrot, but for extra low price. We are going with a whole family to that SuperHiperMarket. Every person can take as many objects, as he/she can carry out from the SuperSale. We have given list of objects with prices and their weight. We also know, what is the maximum weight that every person can stand. What is the maximal value of objects we can buy at SuperSale?

Input Specification

The input consists of *T* test cases. The number of them (1<=*T*<=*1000*) is given on the first line of the input file.

Each test case begins with a line containing a single integer number *N* that indicates the number of objects (*1 <= N <= 1000*). Then follows *N* lines, each containing two integers: P and W. The first integer (1<=P<=100) corresponds to the price of object. The second integer (1<=W<=30) corresponds to the weight of object. Next line contains one integer (1<=G<=100) it's the number of people in our group. Next G lines contains maximal weight (1<=MW<=30) that can stand this i-*th* person from our family (1<=i<=G).

Output Specification

For every test case your program has to determine one integer. Print out the maximal value of goods which we can buy with that family.

Sample Input

```
2

3

72 17

44 23

31 24

1

26

6

64 26

85 22

52 4
```

```
99 18

39 13

54 9

4

23

20

20

26
```

Output for the Sample Input

```
72

514
```

10496 - Collecting Beepers

Time limit: 3.000 seconds

Karel is a robot who lives in a rectangular coordinate system where each place is designated by a set of integer coordinates *(x* and y). Your job is to design a program that will help Karel pick up a number of beepers that are placed in her world. To do so you must direct Karel to the position where each beeper is located. Your job is to write a computer program that finds the length of the shortest path that will get Karel from her starting position, to each of the beepers, and return back again to the starting position.

Karel can only move along the *x* and *y* axis, never diagonally. Moving from one position *(*i, to an adjacent position (i, j + 1), (i, j-1), (i-1, j), or (i+1, j), has a cost of one.

You can assume that Karel's world is never larger than 20 x 20 squares and that there will never be more than 10 beepers to pick up. Each coordinate will be given as a pair *(*x, y) where each value will be in the range 1 through the size of that particular direction of the coordinate system.

Input

First there will be a line containing the number of scenarios you are asked to help Karel in. For each scenario there will first be a line containing the size of the world. This will be given as two integers *(*x-size and y-size). Next there will be one line containing two numbers giving the starting position of Karel. On the next line there will be one number giving the number of beepers. For each beeper there will be a line containing two numbers giving the coordinates of each beeper.

Output

The output will be one line per scenario, giving the minimum distance that Karel has to move to get from her starting position to each of the beepers and back again to the starting position.

Sample Input

```
1
10 10
1 1
4
2 3
5 5
9 4
6 5
```

Sample Output

```
The shortest path has length 24
```

10491 - Cows and Cars

Time limit: 3.000 seconds

In television contests, participants are often asked to choose one from a set of or doors for example, one or several of which lead to different prizes. In this problem we will deal with a specific kind of such a contest. Suppose you are given the following challenge by the contest presenter:

In front of you there are three doors. Two of them hide a cow, the other one hides your prize - a car.
After you choose a door, but before you open it, I will give you an hint, by opening one of the doors which hides a cow (I'll never open the door you have chosen, even if it hides a cow). You will then be able to choose if you want to keep your choice, or if you wish to change to the other unopened door. You will win whatever is behind the door you open.

In this example, the probability you have of winning the car is 2/3 (as hard as it is to believe), assuming you always switch your choice when the presenter gives you the opportunity to do so (after he shows you a door with a cow). The reason of this number (2/3) is this - if you had chosen one of the two cows, you would surely switch to the car, since the presenter had shown you the other cow. If you had chosen the car, you would switch to the remaining cow, therefore losing the prize. Thus, in two out of three cases you would switch to the car. The probability to win if you had chosen to stick with your initial choice would obviously be only 1/3, but that isn't important for this problem.

In this problem, you are to calculate the probability you have of winning the car, for a generalization of the problem above:

- The number of cows is variable

- The number of cars is variable (number of cows + number of cars = total number of doors)

- The number of doors hiding cows that the presenter opens for you is variable (several doors may still be open when you are given the opportunity to change your choice)

You should assume that you always decide to switch your choice to any other of the unopen doors after the presenter shows you some doors with cows behind it.

Input

There are several test cases for your program to process. Each test case consists of three integers on a line, separated by whitespace. Each line has the following format:

NCOWS NCARS NSHOW

Where NCOWS is the number of doors with cows, NCARS is the number of doors with cars and NSHOW is the number of doors the presenter opens for you before you choose to switch to another unopen door.

The limits for your program are:

1 <= NCOWS <= 10000

1 <= NCARS <= 10000

0 <= NSHOW < NCOWS

Output

For each of the test cases, you are to output a line containing just one value - the probability of winning the car assuming you switch to another unopen door, displayed to 5 decimal places.

Sample input

```
2 1 1
5 3 2
2000 2700 900
```

Sample output

```
0.66667
0.52500

0.71056
```