FAU-Programmierwettbewerb 2005

29. Januar 2005

Problemübersicht:

Viel Spaß beim Programmieren!

Die Dokumentation des Java-API ist unter
`http://www2.informatik.uni-erlangen.de/Dokumentation/j2sdk-1.4.2/api/`
einzusehen.

Die Dokumentation der Standard Template Library ist unter
`http://www2.informatik.uni-erlangen.de/Dokumentation/STL/` zu finden.

Problem A
Little Red Riding Hood

Author: Tilmann Spiegelhauer

Time Limit – ??? seconds

You probably all know the fairy tale of Little Red Riding Hood and how she wanted to visit her grandmother, but on her way there met a wolf, finally got eaten by the wolf and was rescued in the end by a brave hunter. Now if Little Red Riding Hood hadn't met the wolf in the first place, she would have avoided a lot of hassle (she wouldn't have been eaten by the wolf and the brave hunter wouldn't have had to rescue her). That she met the wolf in the first place was partly due to her lack of information. She simply didn't know where she might encounter the wolf.

The network of roads connecting Little Red Riding Hood's house and her grandmother's house is a rectangular grid. Little Red Riding Hood's house is situated on the grid's lower left corner and her grandmother's house is at the grid's top right corner. Being particularly bright at math Little Red Riding Hood always wants to get to her grandmother using the shortest possible route. This entails that Little Red Riding Hood can only "move" to left or upwards on the grid. This ensures that she always uses a shortest route. Now if she had known the wolf's possible locations in advance, she might have chosen a path on which she doesn't meet the wolf.

Now given the size of the grid and the wolf's possible locations (weirdly enough the wolf can only meet her at the grid's intersections) compute the number of possible paths from Little Red Riding Hood's house to her grandmother's house under the restriction that Little Red Riding Hood can only move to the left and upwards and **does not** meet the wolf.

Input

There will be several testcases. The grid's width w, $1 \leq w \leq 100$, and grid's height h, $1 \leq h \leq 100$, are on the first line. The last testcase will contain a 0 for height h and a 0 for the width w and should not be processed. In the next line there follows the number $n, 1 \leq n \leq 100$ of the wolf's possible locations. The next n lines contain two integers each. The first denoting the wolf's x, $0 \leq x \leq 100$ coordinate, the second the wolf's y, $0 \leq y \leq 100$ coordinate. Little Red Riding Hood's House is at the point $(0, 0)$ and the grandmother's house is at (w, h). The wolf cannot be at either house.

Output

Output one line for each testcase. If there is more than one path between Little Red Riding Hood's house and the grandmother's house on which Little Red Riding Hood **doesn't** meet the wolf and only moves left and upwards, output the number of paths in the format:

```
There are X paths from Little Red Riding Hood's house to her grandmother's house.
```

If there is exactly one path print:

```
There is one path from Little Red Riding Hood's house to her grandmother's house.
```

Otherwise print:

```
There is no path.
```

The number of paths will always be $\leq 2^{32} - 1$.

Sample Input

Sample Output

Problem B
Cracking your sister's diary

Author: Christian Riess

Time Limit – ??? seconds

There's a certain point in every little sister's life when they decide to hide their most wicked desires in a diary. Since they usually know about your existence and the associated threat that you could read the once written lines, they smartly make an efford to encrypt the text.

But alas, your little sister's knowledge of cryptosystems lacks sophistication, since she only about eight or nine years old, and her own "system" usually has nothing in common with group theory or other intricate mathematical stuff. Consequently, it is more a matter of honour than of resources to crack the code and read the plain text.

Your sister, in particular, has chosen the following (proprietarily developed) crypto system:
The text is always plain text, except the words or sentences that are enclosed by double quotes. In those clauses she left out some letters, which are superfluous if the letters are pronounced as they are pronounced in the German alphabet. See the sample input and output for an easy example.

Input

The input consists of the text of your sister's diary, terminated by EOF. You can assume that your sister uses only lowercase letters, digits and space characters, except the double quotes denoting the beginning and the end of an encrypted clause. There are no umlauts in the text, and every opening double quote is also closed in the same line.

Output

Echo the text outside of the double quotes unchanged and translate the rest by replacing every character by it's "name" in the German alphabet, according to the following table. The double quotes shouldn't appear in the translated text.

a	a	j	jot	s	es	2	zwei
b	be	k	ka	t	te	3	drei
c	ce	l	el	u	u	4	vier
d	de	m	em	v	vau	5	fuenf
e	e	n	en	w	we	6	sechs
f	ef	o	o	x	ix	7	sieben
g	ge	p	pe	y	ypsilon	8	acht
h	ha	q	qu	z	zet	9	neun
i	i	r	er	1	eins	0	null

Sample Input

```
heute gab es ‘‘nt’’
‘‘2fl’’haft ob ‘‘r’’ kommt
unterm dirndel wird ‘‘gjl’’t
```

Sample Output

```
heute gab es ente
zweiefelhaft ob er kommt
unterm dirndel wird gejotelt
```

Problem C
A gentlemen's agreement

Author: Christian Riess

Time Limit – ??? seconds

1962: The nuclear potential of the USSR and the US prompted some military strategists to consider the presumably unpleasant situation of being hit by a intercontinental ballistic missile, if a war between the two parties actually broke out.

One possible solution was an ïdee fixeto appoint a certain area in the world (for instance the Old Europe) where both superpowers and their allies can drop their whole arsenal and look afterwards, who has won. The advantage of such an agreement is obvious: Since Old Europe is neither part of the US nor of the USSR, their own territory would possibly be left unspoiled, but they would have the luxury of an exclusive combat zone for a nice exhaustive war.

The remaining work is a pragmatist's job: Both parties and their allies want the combat zone to be as close as possible, since they have to deploy their troops there, but they want their capitals outside of that zone. Mathematicians suggest that a good location for the battle field minimizes the added quadratic euclidean distance $\sum_{i=1}^{n}(x_i - x_c)^2 + (y_i - y_c)^2$ from every capital to the center of the combat zone. If this optimal battle field is closer than 500 kilometers to one of the capitals, strategists cannot come to a mutual agreement.

Input

In the first line is the number k of testcases. Every testcase starts with a line containing the number n of capitals. The following n lines contain the capital's integer coordinates in a plane (since military maps are always planes). The unit of measure is 1 kilometer.

Output

If the strategists can find a battle field, for instance at $x = 150, y = 20$, output a line containing:

```
Yeah, let's meet at (150/20)!
```

If for instance 2 capitals lie within the optimal battle field, output a line containing:

```
2 nations are displeased by the proposal.
```

If only one capital lies in the combat zone, output a line containing:

```
1 nation is displeased by the proposal.
```

Sample Input

Sample Output

Problem D
The construction company

Author: Tilmann Spiegelhauer

Time Limit – ??? seconds

The mayor of Madman City has decided that the city's crossings need newer and better traffic lights. He has commissioned the company Light's Inc. to remove the old and install the new traffic lights, because it is a very renowned company and one of the leading producers of traffic lights. The people in charge of this task (a mathematician and an engineer) have decided to minimize the ensuing chaos when the traffic lights are replaced, which entails that not all traffic lights can be replaced simultaneously. On the other hand they do not want to replace the traffic lights one at a time as that would take too long. So they decided that if they replaced the traffic lights at one intersection they would not replace the traffic lights at those intersections which were directly connected to the intersection where the traffic light was being replaced.

Now the practical engineer was, of course, interested in the maximum number of intersections where the traffic lights could be replaced simultaneously, so he was looking for the largest set of intersections, where no two intersections in the set are connected by a road. The theoretical mathematician however was interested in the number of different sets of intersections in which no subset containing two intersections is connected by a road and it is not possible to add another road without violating the previous condition.

Input

The first line contains the number of cities which are to be processed. The following lines contain the descrition of the cities. A city consists of a number of intersections and of a number of roads which connect the intersections. The number of intersections $i, 1 \leq i \leq 100$ and the number of roads $r, 1 \leq r \leq 10000$ are on a line. The intersections are numbered from $0...i-1$. The following r lines contain the description of the roads. The description of a road consists of the two intersections it connects. No intersections will be connected directly by two different roads. Furthermore the graph representing the city's network of roads will be connected.

Output

For each each city, output the number of sets of intersections, where no two intersections in the set are connected by a road and it is not possible to add another intersection to the set without violating the first condition, on the first line. Print the number of intersections in the largest set on the second line.

Sample Input

Sample Output

Problem E
Shortest Rings

Author: Thorsten Meinl

Time Limit – 2 seconds

Most organic molecules contain a large number of rings, which are often also fused together. One interesting question in biochemistry and bioinformatics is how many of the rings in a molecule are sufficient to describe all rings in this molecule. These rings are then called a cycle basis, because all other rings can be constructed by joining two or more the the basis members. The rings in a cycle basis are always the shortest rings through a pair of atoms. However the molecule often consists of more rings. In the sample molecule shown on the right there are six rings but only three of them are in the cycle basis. All other rings can be constructed by joining two or more members of the basis. Your task is now to aid the biochemists in determining the cycle basis of molecules.

Input

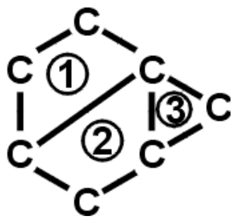
The first line of the input holds the number of molecules that will follow. Then for each molecule first the number of atoms nand then the number of edges m is given in one line separated by a space. The following n lines contain the label of the atoms as a single character followed by a space and a unique index for this atom ranging from 1 to the number of atoms. After that m lines with edges follow. Each line contains the index of the first atom, the type of the bond as a single character (one of -, =, # and :), and the index of the second atom. These three items are separated by single spaces. After that comes the next molecule (or EOF if no more molecules follow).

The maximum number of atoms will not be greater than 1,000, the maximum number of rings in a molecule will not execeed 100.

Output

The output should be a single number in a line for each molecule that is the minimum number of rings in the molecule that form a cycle basis.

Sample Input



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

Sample Output

```
3
```

Problem F
Ultra-QuickSort

Author: Stefan Büttcher

Time Limit – 8 seconds

In this problem, you have to analyze a particular sorting algorithm. The algorithm processes a sequence of n distinct integers by swapping two adjacent sequence elements until the sequence is sorted in ascending order. For the input sequence

```
9 1 0 5 4 ,
```

Ultra-QuickSort produces the output

```
0 1 4 5 9 .
```

Your task is to determine how many swap operations Ultra-QuickSort needs to perform in order to sort a given input sequence.

Input

The input contains several test cases. Every test case begins with a line that contains a single integer $n < 500,000$ – the length of the input sequence. Each of the the following n lines contains a single integer $0 \leq a[i] \leq 999,999,999$, the i-th input sequence element. Input is terminated by a sequence of length $n = 0$. This sequence must not be processed.

Output

For every input sequence, your program prints a single line containing an integer number op, the minimum number of swap operations necessary to sort the given input sequence.

Sample Input

```
5
9
1
0
5
4
3
1
2
3
0
```

Sample Output

```
6
0
```

Problem G
The Same Procedure as Every Year

Author: Dominik Scheder

Time Limit – ??? seconds

It's the same procedure as every year: in early December, the Dominik starts to plan his kick-ass New Year's party. It should be lots of fun: nice people, good food, and plenty of beer. Unfortunately, the Dominik has a problem: during the last year, several couples among his friends broke up, and as you know, it's never a good idea to invite someone and his/her ex-girlfriend/ex-boyfriend at the same time. Consequently, the Dominik won't be able to celebrate New Year's Eve with all his beloved friends. But still, he wishes to have a party with as many people as possible. To make things more complicated, not all his friends are equally important to him. For example, he would rather have his beautiful friend A coming to the party than A's three ex-boyfriends B, C, and D. To formalize things, the Dominik assigns a love value v to each of his friends, where v is a positive integer. Your task is to help the Dominik choosing a feasible subset of his friends (i.e., a subset containing no ex-couples) with which he will be able to celebrate New Year's Eve. Of course, the love value of this set (the sum of all love values of the people in the set) should be as high as possible.

Please note that the Dominik is a very tolerant person and might have gay people among his friends. So don't assume the break-up graph (yes, it's a graph problem, smarty) to be bipartite or anything else. As you might have noted, this problem appears not to be easy. In fact, it is NP-complete. So what should the Dominik do? Should he be content with an approximate solution? No way! But look, since the Dominik doesn't have too many friends anyway, there is still hope for you to solve the problem optimally in a reasonable amount of time. Good luck!

Input

The input consists of several test cases. The first line of each test case contains two integers n, m. n is the numbers of friends the Dominik has, i.e. the number of people among which he can choose whom to invite. Be assured that n will not exceed 20. The second number, m, denotes the number of ex-couples, i.e. the number of pairs of friends the cannot be invited both. Each of the n lines contains a single positive integer v, the value in the k-th of theses lines being the love value of the Dominik's k-th friend. The next m lines each contain two integers $1 \leq a, b \leq n$, telling you that a and b were dating each other some time ago but have broken up. So you cannot have both a and b at the party. The input is ended by a test case where $n, m = 0$, which should not be processed.

Output

For each test case, print the maximal possible love value for this set in a single line.

Sample Input

Sample Output

Problem H
The Bakery

Author: Dominik Scheder

Time Limit – ??? seconds

Suppose you are the owner of a bakery, and your task is to bake several pies for a big wedding in your village. The couple-to-be is already in the church, and all your pies are prepared, but not baked yet. Fortunately, your bakery has more than one oven, so you will be able to bake the pies in parallel. As soon as the last pie is done, you can put them into your van and drive to the church.

Time is short, so you want to be as fast as possible. Note that different pies may take a different time in the oven, and an oven can hold only one pie at a time. Also, all ovens are identical, i.e. a pie takes the same time to bake in any oven.

You want to distribute the pies among the ovens such that the *makespan time*, i.e. the time until the last pie is done, is as small as possible. Since you are not only a baker but also interested in theory of computing, you know that this problem is NP-complete. Unfortunately, the wedding party is really huge and has ordered a gigantic number of pies, so if you are planning to solve the problem optimally, the wedding will already be over, the child will be born and the couple will most probably have been divorced. In other words, nobody will eat your pies then.

Your only chance is to find a schedule that is not optimal, but still rather good, i.e. an approximation. In this problem, you will be asked to find a factor-2 approximation, i.e. a schedule to bake your pies that will not take more than twice the time of the optimal schedule. Be assured that any schedule that satisfies this constraint will be accepted. If you can write a program that finds the optimum efficiently, you will not only be a rich baker, but also a rich computer scientist. But don't try this in this programming contest.

Input

The input consists of several test cases. The first line of each test case contains two integers n, m, denoting the numbers of pies and the numbers of ovens, respectively. The n following lines each contain a single number t, the k-th of them denoting the time pie number k takes. The input ends with a test case where $n, m = 0$, which should not be processed.

Output

For each test case, output m lines, where the k-th line contains the indices of all pies scheduled to be baken in oven k. Your solution will be accepted if your makespan times are not longer than twice the optimal makespan times.

Sample Input

Sample Output

Problem I
Border Crossings

Author: Thorsten Meinl

Time Limit – 2 seconds

In 2300 after several wars the old world finally broke apart. The once united biggest country of the world was split into many parts. Because the regents of the remaining parts wanted to be as separated as possible from their neighbours, they made an agreement that there should be only one road from each part to any of its neighbours. All other roads that crossed the borders were destroyed. Of course they now need border crossing stations on these roads to control the few people that still want to cross the borders. Luckily your father owns the only remaining firm that sells border crossing stations. Of course your father knows all this and wants to start the production as early as possible. Thus he has to calculate the number of border crossing stations the regents will order (they are clever enough to just make one order for all stations because of the price). All he has is a quite recent map of the world with all remaining roads. Your task is now to aid your father in determining the number of border crossing stations that he must produce.

Input

The input will start with the number of maps in the first line. Then the maps follow. Each map is given as a huge quadratic matrix of 0's and 1's. Each line (and the corresponding column) represents a junction of roads. A **1** in line i and column k means that there is a road between the junctions i and k, a **0** means that there is no road.

There will not be more than 10,000 junctions in a map.

Output

The program should output for each given map the number of border crossing stations needed in a single line. Each road that is the only one between two parts of the former old world needs a station.

Sample Input

```
2
0110000000
1001000000
1001000000
0110100000
0001011000
0000101000
0000110100
0000001011
0000000101
0000000110
011000
101001
110000
000011
000101
010110
```

Sample Output

```
2
1
```