Important notes

- Problems X and Y are given out at 9:00.
- Problems A–H are given out at 10:00.
- Your program has to read the problem input from the standard input and write the solution to the standard output.
- Your program has to return 0 as return code.
- End of line is denoted by a single character of code 10, both in the input and the output.
- The last line of the input is terminated by a new line character and you have to output a new line character after the last line of the output.
- The official clock of the contest is the clock of the server. You can see the current time on the submission server webpages. (Do not forget to refresh to see the current time.)
- Due to some possible performance issues, we do not recommend using the `iostream` library in C++.

Problem X: Movies

Introduction

The small town of Unicinema has only one movie theater. The choice of films is very limited: the cinema plays one of 26 possible films every night. If on some day i the cinema plays the same movie as on the previous day $i - 1$, the we say that day i was a boring day. Given the schedule of the movies, your task is to count the number of boring days.

Input

The input contains several blocks of test cases. Each test case consists of a single line containing at most 1000 characters. The characters are between `'A'` and `'Z'` and they denote the sequence of movies played in the cinema.

The input is terminated by a line containing `'0'`.

Output

For each test case, you have to output a single integer on a separate line: the number of boring days.

Sample input

```
ABCD
AABBBCDD
0
```

Sample output

```
0
4
```

Problem Y: Sushi Party

Introduction

You are organizing a party for your friends. To have some food for the evening, you bought some number of sushi rolls. In order to avoid any bad feelings, you want to ensure that everyone gets the same number of sushi rolls (including you), but everyone has to get more than one. For example, if you have 20 sushi rolls, then you can invite 1, 3, 4, or 9 friends. Your task is to write a program that, given the number of sushi rolls, gives you a suggestion on how many friends to invite.

Input

The input contains several blocks of test cases. Each test case consists of a single line containing a single integer $1 \leq n \leq 10000$. The input is terminated by a line containing `'0'`.

Output

For each test case, you have to output a single positive integer on a separate line: a possible number of guests to invite. If it is not possible to invite guests at all for this number of sushi rolls, then output `'No solution.'` (without the quotes).

Sample input

```
12
14
11
20
0
```

Sample output

```
3
6
No solution.
1
```

Problem A: Strange Ordering

Introduction

Let us consider the set of integer numbers between 1 and n inclusive. Let us order them lexicographically (i.e., like in the vocabulary), for example, for $n = 11$ the order would be: 1, 10, 11, 2, 3, 4, 5, 6, 7, 8, 9.

Let us denote the position of the number k in this ordering as $q_{n,k}$. For example, $q_{11,2} = 4$. Given numbers k and m find the smallest n such that $q_{n,k} = m$.

Input

The input contains several blocks of test cases. Each test case consists of a single line containing two integers k and m ($1 \leq k, m \leq 10^9$), separated by a space. The input is terminated by a block with $k = m = 0$.

Output

For each test case, you have to output a single integer on a separate line. If there exists an n such that $q_{n.k} = m$, then write to the output the smallest such n, otherwise write 0.

Sample input

```
2 4
2 1
100000001 1000000000
1000000000 11
0 0
```

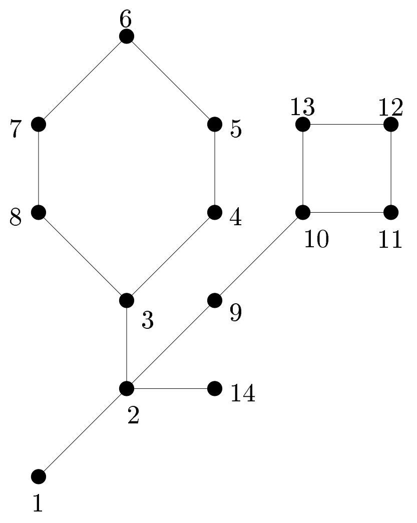
Sample output

```
11
0
100000000888888879
0
```

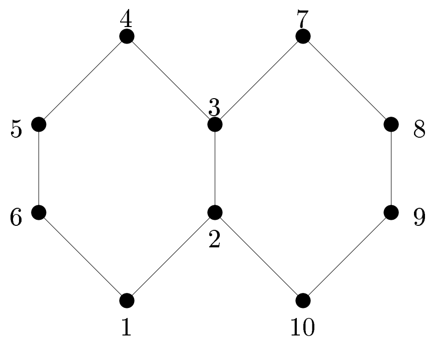
Problem B: Cactus

Introduction

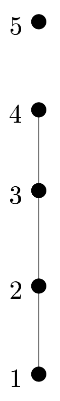
Cactus is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively, cactus is a generalization of a tree where some cycles are allowed. Your task first is to verify if the given graph is a cactus or not. Important difference between a cactus and a tree is that in a cactus we might be able to delete edges such that the resulting subgraph is also a cactus. (Note that a cactus has to be connected, thus the subgraph cannot contain isolated vertices.) The number of such subgraphs (including the graph itself) determines *cactusness* of a graph (this number is one for a cactus that is just a tree). The cactusness of a graph that is not a cactus is considered to be zero.



A cactus



Not a cactus



Not a cactus

The first graph on the picture is a cactus with cactusness 35. The second graph is not a cactus because edge (2, 3) lies on two cycles. The third graph is not a cactus because it is not connected.

Input

The input contains several blocks of test cases. Each test case begins with a line containing two integer numbers n and m ($1 \leq n \leq 20000, 0 \leq m \leq 1000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer number k_i ($2 \leq k_i \leq 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. A path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

The input is terminated by a block with $n = m = 0$.

Output

For each test case in the input, you have to output a single integer on a separate line: the cactusness of the given graph. Note that cactusness can be a *very very* large number.

Sample Input

```
14 3
9 1 2 3 4 5 6 7 8 3
7 2 9 10 11 12 13 10
2 2 14
10 2
7 1 2 3 4 5 6 1
6 3 7 8 9 10 2
5 1
4 1 2 3 4
0 0
```

Sample Output

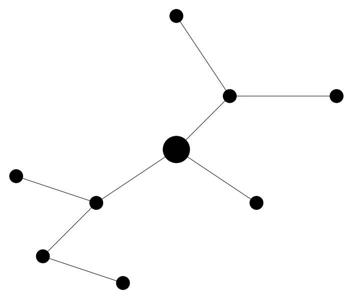
```
35
0
0
```

Problem C: Subway

Introduction

Ultrabandu, the capital city of the Bandulu Kingdom, has a subway system in the form of a tree, i.e. between any pair of stations, there is one and only one way of going by subway. There is a big central station at the Royal Palace. Imagine you are a tourist in Ultrabandu and you want to explore all of the subway system. You start at the central station and pick a subway line at random and jump aboard the subway car. Every time you arrive at a station, you pick one of the subway lines that leads to a station that you have not visited. If there is none left to explore at your current station, you take the subway line back on which you first came to the station, until you eventually have traveled along all of the lines twice, once for each direction. At that point you are back at the central station. Afterwards, all you remember of the order of your exploration is whether you went further away from the central station or back towards it at any given time, i.e. you could encode your tour as a binary string, where 0 encodes taking a subway line getting you one station further away from the central station, and 1 encodes getting you one station closer to the central station.

After a long day of exploring the city, you relax at one of the fine pubs in Ultrabandu. You are drinking beer with another tourist who also spent the day exploring the subway system. Somewhat surprisingly, he encoded his tour the same way as you: 0 means going away from the central station, 1 means going closer to the central station. However, he obtained a different sequence of 0's and 1's than you. Did one of you make a mistake in the encoding, or is it possible that the two different sequences encode the same subway system? Your task is to write a program that decides whether the two sequences encode the same subway system or not.



0010011101001011
0100011011001011
0100101100100111

To the left: A subway tree system. The larger dot is the central station. To the right: Three out of several possible encodings of exploration tours for the subway system.

Input

On the first line of input is a single positive integer n, telling the number of test scenarios to follow. Each test scenario consists of two lines, each containing a string of the characters '0' and '1' of length at most 3000, both describing a correct exploration tour of a subway tree system.

Output

For each test scenario, output one line containing the text `'same'` (without quotes) if the two strings may encode exploration tours of the same subway tree system, or the text `'different'` if the two strings cannot be exploration tours of the same subway tree system.

Sample input

```
2
0010011101001011
0100011011001011
0100101100100111
0011000111010101
```

Sample input

```
same
different
```

Problem D: IP Networks

Introduction

Alex is administrator of IP networks. His clients have a bunch of individual IP addresses and he decided to group all those IP addresses into the smallest possible IP network. Each IP address is a 4-byte number that is written byte-by-byte in a decimal dot-separated notation "byte0.byte1.byte2.byte3" (quotes are added for clarity). Each byte is written as a decimal number from 0 to 255 (inclusive) without extra leading zeroes.

IP network is described by two 4-byte numbers—network address and network mask. Both network address and network mask are written in the same notation as IP addresses. In order to understand the meaning of network address and network mask you have to consider their binary representation. Binary representation of IP address, network address, and network mask consists of 32 bits: 8 bits for byte0 (most significant to least significant), followed by 8 bits for byte1, followed by 8 bits for byte2, and followed by 8 bits for byte3.

IP network contains a range of 2^n IP addresses where $0 \le n \le 32$. Network mask always has $32 - n$ first bits set to one, and n last bits set to zero in its binary representation. Network address has arbitrary $32 - n$ first bits, and n last bits set to zero in its binary representation. IP network contains all IP addresses whose $32 - n$ first bits are equal to $32 - n$ first bits of network address with arbitrary n last bits. We say that one IP network is smaller than the other IP network if it contains fewer IP addresses. For example, IP network with network address 194.85.160.176 and network mask 255.255.255.248 contains 8 IP addresses from 194.85.160.176 to 194.85.160.183 (inclusive).

Input

The input contains several blocks of test cases. Each case begins with a line containing a single integer $1 \le m \le 1000$. The following m lines contain IP addresses, one address on a line. An IP address may appear more than once in the input file.

The input is terminated by a block with $m = 0$.

Output

For each test case, you have to output two lines that describe the smallest possible IP network that contains all IP addresses from the input file. Write network address on the first line and network mask on the second line.

Sample Input

```
3
194.85.160.177
194.85.160.183
194.85.160.178
0
```

Sample Output

```
194.85.160.176
255.255.255.248
```

Problem E: Folding

Introduction

Bill is trying to compactly represent sequences of capital alphabetic characters from `'A'` to `'Z'` by folding repeating subsequences inside them. For example, one way to represent a sequence `AAAAAAAAAABABABCCD` is `10(A)2(BA)B2(C)D`. He formally defines folded sequences of characters along with the unfolding transformation for them in the following way:

- A sequence that contains a single character from `'A'` to `'Z'` is considered to be a folded sequence. Unfolding of this sequence produces the same sequence of a single character itself.
- If S and Q are folded sequences, then SQ is also a folded sequence. If S unfolds to S' and Q unfolds to Q', then SQ unfolds to $S'Q'$.
- If S is a folded sequence, then $X(S)$ is also a folded sequence, where X is a decimal representation of an integer number greater than 1. If S unfolds to S', then $X(S)$ unfolds to S' repeated X times.

According to this definition it is easy to unfold any given folded sequence. However, Bill is much more interested in the reverse transformation. He wants to fold the given sequence in such a way that the resulting folded sequence contains the least possible number of characters.

Input

The input file contains several test cases. Each test case consists of a single line of characters from `'A'` to `'Z'` with at least 1 and at most 200 characters.

The input is terminated by a line containing `'0'`.

Output

For each test case, output a single line that contains the shortest possible folded sequence that unfolds to the sequence that is given in the input file. If there are many such sequences then write any one of them.

Sample Input

```
AAAAAAAAAABABABCCD
MAKMAKYESYESYESMAKMAKYESYESYES
0
```

Sample Output

```
9(A)3(AB)CCD
2(2(MAK)3(YES))
```

Problem F: Pearls

Introduction

In Pearlania everybody is fond of pearls. One company, called The Royal Pearl, produces a lot of jewelry with pearls in it. The Royal Pearl has its name because it delivers to the royal family of Pearlania. But it also produces bracelets and necklaces for ordinary people. Of course the quality of the pearls for these people is much lower then the quality of pearls for the royal family. In Pearlania pearls are separated into 100 different quality classes. A quality class is identified by the price for one single pearl in that quality class. This price is unique for that quality class and the price is always higher then the price for a pearl in a lower quality class.

Every month the stock manager of The Royal Pearl prepares a list with the number of pearls needed in each quality class. The pearls are bought on the local pearl market. Each quality class has its own price per pearl, but for every complete deal in a certain quality class one has to pay an extra amount of money equal to ten pearls in that class. This is to prevent tourists from buying just one pearl.

The Royal Pearl is suffering from the slowdown of the global economy. Therefore the company needs to be more efficient. The CFO (chief financial officer) has discovered that he can sometimes save money by buying pearls in a higher quality class than is actually needed. No customer will blame The Royal Pearl for putting better pearls in the bracelets, as long as the prices remain the same. For example 5 pearls are needed in the 10 Euro category and 100 pearls are needed in the 20 Euro category. That will normally cost: $(5 + 10) \cdot 10 + (100 + 10) \cdot 20 = 2350$ Euro. Buying all 105 pearls in the 20 Euro category only costs: $(5 + 100 + 10) \cdot 20 = 2300$ Euro.

The problem is that it requires a lot of computing work before the CFO knows how many pearls can best be bought in a higher quality class. You are asked to help The Royal Pearl with a computer program. Your task is to write a program that, given a list with the number of pearls and the price per pearl in different quality classes, give the lowest possible price needed to buy everything on the list. Pearls can be bought in the requested, or in a higher quality class, but not in a lower one.

Input

The input contains several blocks of test cases. Each test case starts with a line containing the number of categories $1 \leq c \leq 100$. Then, c lines follow, each with two numbers a_i and p_i. The first of these numbers is the number of pearls a_i needed in a class ($1 \leq a_i \leq 1000$). The second number is the price per pearl p_i in that class ($1 \leq p_i \leq 1000$). The qualities of the classes (and so the prices) are given in ascending order. All numbers in the input are integers.

The input is terminated by a block with $c = 0$.

Output

For each test case, output a single line containing a single number: the lowest possible price needed to buy everything on the list.

Sample input

```
2
100 1
100 2
3
1 10
1 11
100 12
0
```

Sample input

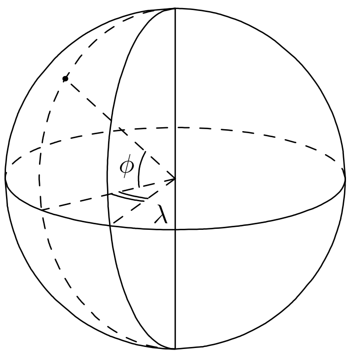
```
330
1344
```

Problem G: World Destruction

Introduction

The mad scientist Doctor Factor is devising a plan to destroy the world. The plan is to place a number of high-yield nuclear bombs in different places around the world, so that their simultaneous detonation will destroy everything on the planet. For the purpose of planning this operation, bombs have destruction distance—the distance from the point of bomb's detonation to all the places where everything is considered destroyed.

Places for the bombs have been already selected. Now Doctor Factor wants to minimize the cost of production for the bombs. It is expensive to design atomic bombs tailored for different destruction distances, so the idea is to design a bomb with a specific destruction distance and to produce bombs according to this design for all the selected places. The problem is to find the minimal required destruction distance for the bomb design, so that destruction of the whole world is ensured.



For this problem the world is modeled as a sphere of a unit radius. The coordinates of the selected points for the bombs are specified in geographic coordinate system with latitude ϕ ($-90° < \phi < 90°$) and longitude λ ($-180° < \lambda \leq 180°$). Latitude is the angle between a point and the equator, and longitude is the angle between a point and the prime meridian. The bombs are never placed on the poles, so their latitude is always less than 90 degrees by its absolute value.

Distances for destruction purposes are measured on the sphere. For example, the distance between the poles is exactly π. The world is considered destroyed if the distance from any point on the sphere to the closest bomb is less or equal to the destruction radius.

Input

The input contains several blocks of test cases. The first line of the input file contains a single integer number n ($1 \leq n \leq 20$), the number of the bombs. The following n lines describe the places of the bombs. Each line contains two integer numbers ϕ_i and λ_i ($-90 < \phi < 90, -180 < \lambda \leq 180$), latitude and longitude of the bomb. No two bombs are situated in the same place.

The input is terminated by a block with $n = 0$.

Output

For each test case, output a single number: the minimal destruction radius of the bombs that ensures destruction of the whole world. The answer must be precise up to 10^{-6}.

Sample Input

```
4
59 30
53 83
41 69
41 41
0
```

Sample Output

```
2.864479
```

Problem H: Tourist Programmes

Introduction

City Tourist Office offers several sightseeing programmes in the city. Each attractive place in the city is located at a street junction. The route of each programme starts at the central junction and visits attractive places and finally returns to the central junction. For efficiency reason, any programme route visits any junction at most once. The office offers any possible such round trip for programme. Tourist groups can walk in both directions in any street, but the structure of the streets is very special, because any junction is the junction of either two or four streets. Because of this structure, it may happen, that some attractive place can not be visited by any programme.

You are to write a program that computes the street junctions that can *not* be visited by any programme.

Input

The input contains several blocks of test cases. The first line of each case contains two integers n and m, where n ($3 \leq n \leq 10000$) is the number of street junctions and m is the number of streets. The street junctions are identified by the integers from 1 to n. The identifier of the central junction is 1. Each of the next m lines contains two integers p and q, that means that there is a street between junction p and q. For any two different junctions there is at most one street between them.

The input is terminated by a block with $n = 0$ and $m = 0$.

Output

For each test case in the input, you have to output two lines. The first line must contain a single integer k, the number of junctions that are not contained in any programme. The second line should contain k numbers separated by a single space, the identifiers of the junctions in ascending order. If $k = 0$, then the second line must be an empty line.

Sample input

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

Sample input

```
0

2
5 6
```