Maratona de Programação UFPR

- This problemset contains six problems and is nine pages long.
- The input is redirected to the standard input (stdin) and the output is read from the standard output (stdout).
- Each problem will be tested with a single file as the input. Pay attention to the problem specification regarding how the problem input ends.
- The original statements were taken from a Russian website. The grammar and choice of words is not flawless, but **do not change the intended meaning of the sentences**.

A: Free Cash

File: cash.[pas|c|cpp|java]

Valera runs a 24/7 fast food cafe. He magically learned that next day n people will visit his cafe. For each person we know the arrival time: the i-th person comes exactly at h_i hours m_i minutes. The cafe spends less than a minute to serve each client, but if a client comes in and sees that there is no free cash, than he doesn't want to wait and leaves the cafe immediately.

Valera is very greedy, so he wants to serve all n customers next day (and get more profit). However, for that he needs to ensure that at each moment of time the number of working cashes is no less than the number of clients in the cafe.

Help Valera count the minimum number of cashes to work at his cafe next day, so that they can serve all visitors.

Input

The input contains several test cases. Each test case starts with a line containing an integer n ($1 \le n \le 10^5$), that is the number of cafe visitors. Each of the following n lines has two space-separated integers h_i and m_i ($0 \le h_i \le 23; 0 \le m_i \le 59$), representing the time when the i-th person comes into the cafe. Note that the time is given in the chronological order. All time is given within a 24-hour period.

The last test case is followed by a line containing a single zero.

Output

For each test case print a single integer — the minimum number of cashes, needed to serve all clients next day.

Sample Input	Sample Output
4 8 0 8 10 8 10 8 45 3 0 12 10 11 22 22 0	2 1

Note

In the first sample it is not enough one cash to serve all clients, because two visitors will come into cafe in 8:10. Therefore, if there will be one cash in cafe, then one customer will be served by it, and another one will not wait and will go away.

In the second sample all visitors will come in different times, so it will be enough one cash.

B: Lucky Substring

File: lucky.[pas|c|cpp|java]

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits 4 *and* 7*. For example, numbers* 47, 744, 4 *are lucky and* 5, 17, 467 *are not.*

One day Petya was delivered a string s, containing only digits. He needs to find a string that:

- represents a lucky number without leading zeroes,
- is not empty,
- is contained in s as a substring the maximum number of times.

Among all the strings for which the three conditions given above are fulfilled, Petya only needs the lexicographically minimum one. Find this string for Petya.

Input

The input contains several test cases. Each test case contains a single line with a non-empty string s whose length can range from 1 to 50, inclusive. The string only contains digits. The string can contain leading zeroes.

The last test case is followed by a line containing a single zero. There will not be any test case consisting of only one zero.

Output

For each test case print one line with the answer to Petya's problem. If the sought string does not exist, print "-1" (without quotes).

Sample Input	**Sample Output**
047 16 472747 0	4 -1 7

Note

The lexicographical comparison of strings is performed by the $\leq$ operator in the modern programming languages. String x is lexicographically less than string y either if x is a prefix of y, or exists such i $(1 \leq i \leq min(|x|, |y|))$, that $x_i \leq y_i$ and for any j $(1 \leq j \leq i)$ $x_j = y_j$. Here $|a|$ denotes the length of string a.

In the first sample three conditions are fulfilled for strings "4", "7" and "47". The lexicographically minimum one is "4".

In the second sample s has no substrings which are lucky numbers.

In the third sample the three conditions are only fulfilled for string "7".

C: Shifts

`File: shifts.[c|cpp|java]`

You are given a table consisting of n rows and m columns. Each cell of the table contains a number, 0 or 1. In one move we can choose some row of the table and cyclically shift its values either one cell to the left, or one cell to the right.

To *cyclically shift* a table row one cell to the right means to move the value of each cell, except for the last one, to the right neighboring cell, and to move the value of the last cell to the first cell. A cyclical shift of a row to the left is performed similarly, but in the other direction. For example, if we cyclically shift a row "00110" one cell to the right, we get a row "00011", but if we shift a row "00110" one cell to the left, we get a row "01100".

Determine the minimum number of moves needed to make some table column consist only of numbers 1.

Input

The input contains several test cases. In each test case the first line contains two space-separated integers: n ($1 \le n \le 100$) - the number of rows in the table and m ($1 \le m \le 10^4$) - the number of columns in the table. Then n lines follow, each of them contains m characters "0" or "1": the j-th character of the i-th line describes the contents of the cell in the i-th row and in the j-th column of the table. It is guaranteed that the description of the table contains no other characters besides "0" and "1".

The last test case is followed by a line containing two zeros.

Output

For each test case print a single number: the minimum number of moves needed to get only numbers 1 in some column of the table. If this is impossible, print "-1" (without quotes).

Sample Input	**Sample Output**
3 6 101010 000100 100000 2 3 111 000 0 0	3 -1

Note

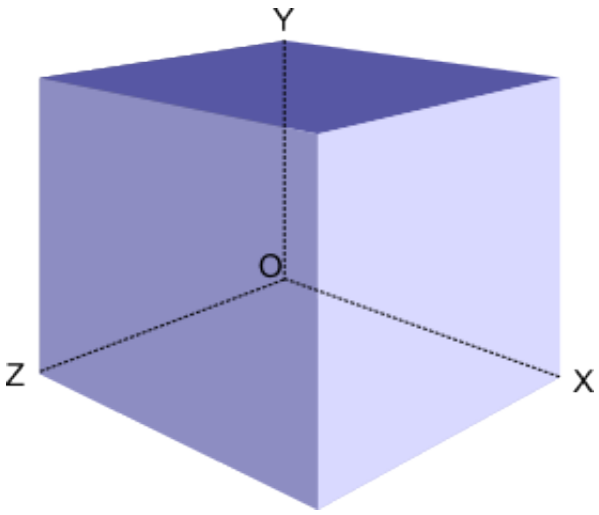
In the first sample one way to achieve the goal with the least number of moves is as follows: cyclically shift the second row to the right once, then shift the third row to the left twice. Then the table column before the last one will contain only 1s.

In the second sample one can't shift the rows to get a column containing only 1s.

D: Magic Box

`File: box.[c|cpp|java]`

One day Vasya was going home when he saw a box lying on the road. The box can be represented as a rectangular parallelepiped. Vasya needed no time to realize that the box is special, as all its edges are parallel to the coordinate axes, one of its vertices is at point $(0, 0, 0)$, and the opposite one is at point (x_1, y_1, z_1). The six faces of the box contain some numbers $a_1, a_2, \ldots, a_6$, exactly one number right in the center of each face.



The numbers are located on the box like that:

- number a_1 is written on the face that lies on the ZOX plane;
- a_2 is written on the face, parallel to the plane from the previous point;a
- a_3 is written on the face that lies on the XOY plane;
- a_4 is written on the face, parallel to the plane from the previous point;
- a_5 is written on the face that lies on the YOZ plane;
- a_6 is written on the face, parallel to the plane from the previous point.

At the moment Vasya is looking at the box from point (x, y, z). Find the sum of numbers that Vasya sees. Note that all faces of the box are not transparent and Vasya can't see the numbers through the box. The picture contains transparent faces just to make it easier to perceive. You can consider that if Vasya is looking from point, lying on the plane of some face, than he can not see the number that is written on this face. It is enough to see the center of a face to see the corresponding number for Vasya. Also note that Vasya always reads correctly the a_i numbers that he sees, independently of their rotation, angle and other factors (that is, for example, if Vasya sees some $a_i = 6$, then he can't mistake this number for 9 and so on).

Input

The input contains several test cases. In each test case the fist input line contains three space-separated integers x, y and z ($|x|, |y|, |z| \leq 10^6$) - the coordinates of Vasya's position in space. The second line contains three space-separated integers x_1, y_1, z_1 ($1 \leq x_1, y_1, z_1 \leq 10^6$) - the coordinates of the box's vertex that is opposite to the vertex at point $(0, 0, 0)$. The third line contains six space-separated integers $a_1, a_2, \ldots, a_6$ ($1 \leq a_i \leq 10^6$) the numbers that are written on the box faces.

It is guaranteed that point (x, y, z) is located strictly outside the box.

The last test case is followed by a line containing three zeros.

Output

For each test case print a single integer - the sum of all numbers on the box faces that Vasya sees.

Sample Input	**Sample Output**
2 2 2 1 1 1 1 2 3 4 5 6 0 0 10 3 2 3 1 2 3 4 5 6 0 0 0	12 4

Note

The first sample corresponds to perspective, depicted on the picture. Vasya sees numbers a_2 (on the top face that is the darkest), a_6 (on the right face that is the lightest) and a_4 (on the left visible face).

In the second sample Vasya can only see number a_4.

E: Spiders

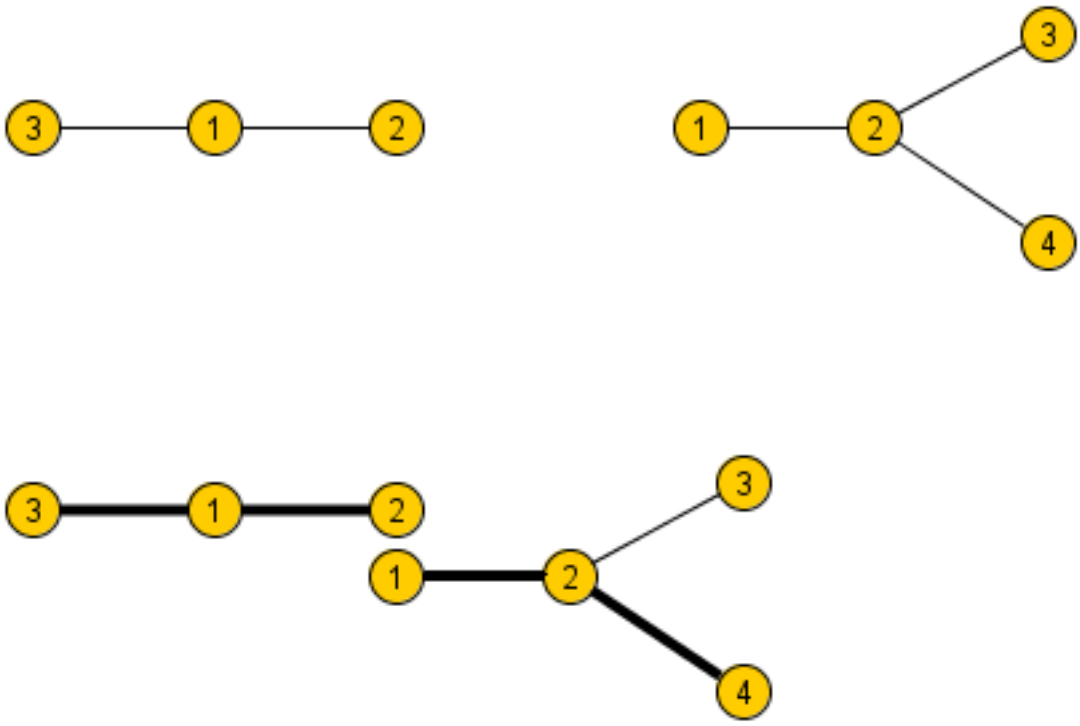
`File: spiders.[c|cpp|java]`

One day mum asked Petya to sort his toys and get rid of some of them. Petya found a whole box of toy spiders. They were quite dear to him and the boy didn't want to throw them away. Petya conjured a cunning plan: he will glue all the spiders together and attach them to the ceiling. Besides, Petya knows that the lower the spiders will hang, the more mum is going to like it and then she won't throw his favourite toys away. Help Petya carry out the plan.

A spider consists of k beads tied together by $k-1$ threads. Each thread connects two different beads, at that any pair of beads that make up a spider is either directly connected by a thread, or is connected via some chain of threads and beads.

Petya may glue spiders together directly gluing their beads. The length of each thread equals 1. The sizes of the beads can be neglected. That's why we can consider that gluing spiders happens by identifying some of the beads (see the picture). Besides, the construction resulting from the gluing process should also represent a spider, that is, it should have the given features.

After Petya glues all spiders together, he measures the length of the resulting toy. The distance between a pair of beads is identified as the total length of the threads that connect these two beads. The length of the resulting construction is the largest distance between all pairs of beads. Petya wants to make the spider whose length is as much as possible.



The picture two shows two spiders from the second sample. We can glue to the bead number 2 of the first spider the bead number 1 of the second spider. The threads in the spiders that form the sequence of threads of maximum lengths are highlighted on the picture.

Input

The input contains several test cases. In each test case the first line of input contains one integer n ($1 \leq n \leq 100$) - the number of spiders. Next n lines contain the descriptions of each spider: integer n_i ($2 \leq n_i \leq 100$) - the number of beads, then $n_i - 1$ pairs of numbers denoting the numbers of the beads connected by threads. The beads that make up each spider are numbered from 1 to n_i.

The last test case is followed by a line containing a single zero.

Output

For each test case print a single number - the length of the required construction.

Sample Input	**Sample Output**
1 3 1 2 2 3 2 3 1 2 1 3 4 1 2 2 3 2 4 2 5 1 2 2 3 3 4 3 5 7 3 4 1 2 2 4 4 6 2 7 6 5 0	2 4 7

F: Watermelon

File: watermelon.[c|cpp|java]

One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.

Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

Input

The input contains several test cases. In each test case the first input line contains a integer number w ($1 \leq w \leq 100$) - the weight of the watermelon bought by the boys.

The last test case is followed by a line containing a single zero.

Output

For each test case print "YES", if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and "NO" in the opposite case.

Sample Input	**Sample Output**
7 8 2 6 0	NO YES NO YES

Note

For example, the boys can divide the 8 kg watermelon into two parts of 2 and 6 kilos respectively (another variant — two parts of 4 and 4 kilos).