Problem A. To the Max

Time limit 1000 ms
Mem limit 10000 kB

Given a two-dimensional array of positive and negative integers, a sub-rectangle is any contiguous sub-array of size 1*1 or greater located within the whole array. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the maximal sub-rectangle.
As an example, the maximal sub-rectangle of the array:

0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
is in the lower left corner:

9 2
-4 1
-1 8
and has a sum of 15.

Input
The input consists of an N * N array of integers. The input begins with a single positive integer N on a line by itself, indicating the size of the square two-dimensional array. This is followed by N^2 integers separated by whitespace (spaces and newlines). These are the N^2 integers of the array, presented in row-major order. That is, all numbers in the first row, left to right, then all numbers in the second row, left to right, etc. N may be as large as 100. The numbers in the array will be in the range [-127,127].

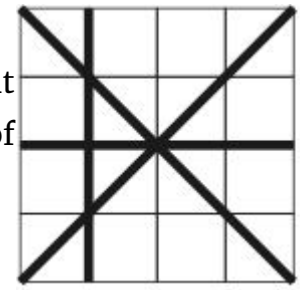
Output
Output the sum of the maximal sub-rectangle.

Sample

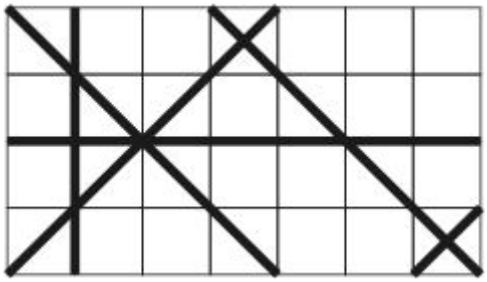
Input	Output
4 0 -2 -7 0 9 2 -6 2 -4 1 -4 1 -1 8 0 -2	15

Problem B. Inlay Cutters

Time limit 1000 ms
Mem limit 65536 kB



The factory cuts rectangular M × N granite plates into pieces using a special machine that is able to perform cuts in 4 different directions: vertically, horizontally, and diagonally at the angle of 45 degrees to the sides of the plate. Every cut is a straight line that starts and ends on the side of the plate.



The factory has been ordered to produce tiles for the inlay, each tile of which is a 45 degrees right triangle. To reduce the time to deliver the tiles it was decided to take all triangles from the already cut plates. Information about all performed cuts is available and your task is to compute the number of triangles of any size that were produced.

Input

The input describes the cuts that were performed on a single rectangular plate. The first line of the input file contains three integer numbers M, N, and K, separated by spaces. M and N ($1 \le M, N \le 50$) are the dimensions of the plate, and K ($0 \le K \le 296$) is the number of cuts. Next K lines describe the cuts. ith cut is described by four integer numbers $X_{i,1}$, $Y_{i,1}$, $X_{i,2}$, and $Y_{i,2}$, separated by spaces, that represent the starting and ending point of the cut. Both starting ($X_{i,1}$, $Y_{i,1}$) and ending ($X_{i,2}$, $Y_{i,2}$) points of the cut are situated on the plate's border. Both points of the cut are different and the cut goes through the plate. Here, the coordinates by the X axis run from 0 to M, and the coordinates by the Y axis run from 0 to N. All cuts are different.

Output

Write to the output a single integer number - the number of triangles that were produced by the cuts.

Sample

Input	Output
7 4 6 6 0 7 1 1 4 1 0 0 4 4 0 0 0 4 4 0 2 7 2 7 0 3 4	8

Problem C. Frobenius

Time limit	1000 ms
Mem limit	65536 kB
OS	Linux

The *Frobenius problem* is an old problem in mathematics, named after the German mathematician G. Frobenius (1849–1917).

Let $a_1, a_2, \ldots, a_n$ be integers larger than 1, with greatest common divisor (gcd) 1. Then it is known that there are finitely many integers larger than or equal to 0, that cannot be expressed as a linear combination $w_1a_1 + w_2a_2 + \ldots + w_na_n$ using integer coefficients $w_i \geq 0$. The largest of such nonnegative integers is known as the Frobenius number of $a_1, a_2, \ldots, a_n$ (denoted by $F(a_1, a_2, \ldots, a_n)$). So: $F(a_1, a_2, \ldots, a_n)$ is the largest nonnegative integer that cannot be expressed as a nonnegative integer linear combination of $a_1, a_2, \ldots, a_n$.

For $n = 2$ there is a simple formula for $F(a_1, a_2)$. However, for $n \geq 3$ it is much more complicated. For $n = 3$ only for some special choices of a_1, a_2, a_3 formulas exist. For $n > 4$ no formulas are known at all.

We will consider here the Frobenius problem for $n = 4$. In this case our version of the problem can be formulated as follows. Let four integers a, b, c and d be given, with $a, b, c, d > 1$ and $\gcd(a, b, c, d) = 1$. We want to know two things.

- How many nonnegative integers less than or equal to 1,000,000 cannot be expressed as a nonnegative integer linear combination of the values a, b, c and d?
- Is the Frobenius number of a, b, c and d less than or equal to 1,000,000 and if so, what is its value?

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line, containing four integers a, b, c, d (with $1 < a, b, c, d \leq 10,000$ and $\gcd(a, b, c, d) = 1$), separated by single spaces.

Output

For every test case in the input file, the output should contain two lines.

- The first line contains the number of integers between 0 and 1,000,000 (boundaries included) that cannot be expressed as $a \cdot w + b \cdot x + c \cdot y + d \cdot z$, where w, x, y, z are nonnegative (meaning ≥ 0) integers.
- The second line contains the Frobenius number if this is less than or equal to 1,000,000 and otherwise -1, meaning that the Frobenius number of a, b, c and d is larger than 1,000,000.

Sample

Input	Output
3 8 5 9 7 5 8 5 5 1938 1939 1940 1937	6 11 14 27 600366 −1

Problem D. Old Wine Into New Bottles

Time limit 1000 ms
Mem limit 65536 kB

Wine bottles are never completely filled: a small amount of air must be left in the neck to allow for thermal expansion and contraction. If too little air is left in the bottle, the wine may expand and expel the cork; if too much air is left in the bottle, the wine may spoil. Thus each bottle has a minimum and maximum capacity.
Given a certain amount of wine and a selection of bottles of various sizes, determine which bottles to use so that each is filled to between its minimum and maximum capacity and so that as much wine as possible is bottled.

Input
The first line of input contains two integers: the amount of wine to be bottled (in litres, between 0 and 1,000,000) and the number of sizes of bottles (between 1 and 100). For each size of bottle, one line of input follows giving the minimum and maximum capacity of each bottle in millilitres. The maximum capacity is not less than 325 ml and does not exceed 4500 ml. The minimum capacity is not less than 95% and not greater than 99% of the maximum capacity. You may assume that an unlimited number of each bottle is available.

Output
Your output should consist of a single integer: the amount of wine, in ml, that cannot be bottled.

Sample

Input	Output
10 2 4450 4500 725 750	250

Hint
Sample Input (2)

```
10000 2


4450 4500


725 750
```

Output for Sample Input (2)

```
0
```

Problem E. Super Assassin

Time limit	10000 ms
Case time limit	2000 ms
Mem limit	65536 kB
OS	Linux

The Nerubian Assassin, Anub'arak, is a notorious hero in the world of DotA. He(It?) might not be a good warrior, but his terrific assaulting abilities can always frighten his enemies. Recently Anub'arak had received an order to kill a very important person. The protection is almost invulnerable, so Anub'arak must make full use of his abilities and try to knock that guy down in several seconds. Due to some fantastic reasons, you are standing at the same side of Anub'arak and you must make a perfect plan for him now.

Anub'arak have N attacking skills, when the ith skill was used alone, it will deal D_i damage instantly. Besides, most of Anub'arak 's skill has a powerful effect which can enhance the damage of next strike to a higher level----The skill with an enhancing ability of E_i will produce a (Ei × 10) percent extra damage for the next strike. A skill with Ei = 0 has no enhancing effect. Each of these skills can be used at most once or the enemy will have a chance to counterattack. Since time is not

enough, Anub'arak can select at most 6 skills, and your task is to calculate the maximum possible damage. It is guaranteed that the result is a 32-bit integer.

Input

The first line of the input file contains a number N ($1 \le N \le 300$), which stands for the total skills Anub'arak can use. The following N lines contain two integers each. The first integer D_i ($10 \le D_i \le 100,000,000$, which is a multiple of 10) describes the basic damage of the ith skill, and the second integer E_i ($0 \le E_i \le 10$) describe the enhancing level of the ith skill.

Output

Only a integer, which stands for the possible maximum damage.

Sample

Input	Output
7 700 7 600 6 500 5 400 4 300 3 200 2 100 1	3940

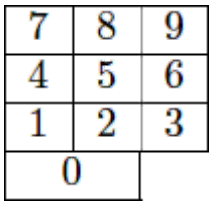
Hint

The sequence of skills is: (200,2)->(400,4)->(600,6)->(700,7)->(500,5)->(300,3)
Total damage is:200+400*1.2+600*1.4+700*1.6+500*1.7+300*1.5=3940

Problem F. RSI

Time limit	1000 ms
Mem limit	65536 kB
OS	Linux

You have the goal of becoming the world's fastest two-fingered typist. In this problem, your goal is to optimize the movement of your fingers when typing numeric values in order to ensure that you finish typing a number in the shortest amount of time possible. Your numeric keyboard has the following layout:



For convenience, we refer to the cells above according to their row and column; hence, the "5" key is at position (2, 2), and the "0" key takes up both positions (4, 1) and (4, 2). At time 0, your left pointer finger is on the "4" key and your right pointer finger is on the "5" key. During each time interval, each finger may press the key underneath it, move vertically one position, or move horizontally one position. Although both fingers may move simultaneously within a single time interval,

- at most one key may be pressed during any given time interval,
- the left pointer finger's column must always be less than the right pointer finger's column at the end of each time interval, and
- both fingers must always be above one of the 10 keys in the diagram at the end of each time interval (e.g., neither finger cannot hover over position (4, 3)).
- The "0" key may be pressed by a finger at either positions (4, 1) or (4, 2).

Examples

- Typing "56" takes three time units. At time 1, both left and right fingers have moved one position to the right and are on keys "5" and "6" respectively. Then, each key is pressed sequentially.
- Typing "71" takes five time units. During the first two time units, the left finger moves up to the "7" and presses the key. However, the right finger is not allowed to be in the same column as the left finger, and hence the left finger takes two time units to get to the "1" key and one time unit to press it.

Input

The input test file will contain multiple test cases. Each test case consists of a single line containing a string of between 1 and 100 digits. The end-of-file is marked by a line containing the word “eof” and should not be processed.

Output

For each input case, output the minimum number of time units required to type the given digits.

Sample

Input	Output
56 71 902 eof	3 5 6

Problem G. Double Trouble

Time limit 5000 ms
Mem limit 10000 kB

Alice Catherine Morris and her sister Irene Barbara frequently send each other e-mails. Ever wary of interceptions and wishing to keep their correspondence private, they encrypt their messages in two steps. After removing all nonalphabetic characters and converting all letters to upper case, they: 1)replace each letter by the letter s positions after it in the alphabet (1 <= s <= 25)|we call this a shift by s-and then, 2) divide the result of step 1 into groups of m letters and reverse the letters in each group (5 <= m <= 20). If the length of the message is not divisible by m, then the last k (less than m) letters are reversed. For example, suppose s = 2 and m = 6. If the plaintext were
Meet me in St. Louis, Louis.
after removing unwanted characters and changing to upper case we get
MEETMEINSTLOUISLOUIS
We will call this the modified plaintext. We then shift each letter by 2 (Y would be replaced with A and Z would be replaced by B, here), getting the intermediate result:
OGGVOGKPUVNQWKUNQWKU
And finally reverse every group of 6 letters:
GOVGGOQNVUPKWQNUKWUK
Note the last two letters made up the last reversed group. As is customary, we write the result in groups
of 5 letters. So the ciphertext would be:
GOVGG OQNVU PKWQN UKWUK
Alas, it's not so hard to find the values for s and m when the ciphertext is intercepted. In fact it's even easier if you know a crib, which is a word in the modified plaintext. In the above example, LOUIS would be a crib. Your job here is to find s and m when presented with a ciphertext and a crib.

Input
Input will consist of multiple problem instances. The first line of input will contain a positive integer indicating the number of problem instances. The input for each problem will consist of multiple lines. The first line of input for a problem will contain the integer n (20 <= n <= 500) which is equal to the number of characters in the ciphertext. The following lines will contain the ciphertext, all upper case in groups of 5 letters separated by a single space. (The last group of letters may contain fewer than 5 letters.) There will be 10 groups of letters per line, except possibly for the last line of ciphertext. The input line following the last line of ciphertext will contain the crib; a single word consisting of between 4 and 10 (inclusive) upper case characters.

Output

Output will be two integers, s and m on a line, separated by a single space, indicating the encryption key that produces the crib, where s is the shift and m is the reversed group size. If there is more than one solution, output the one with smallest s. If there is more than one with the same s, output the one with smallest m. If no such s and m exist, output the message Crib is not encrypted.

Sample

Input	Output
4 83 FIQMF IISFN QMFIB EOPFH FNQMV PSFIU IZNGP UPEUS BFPEP PEPPE PPEPN QMFIP EOPIS FIQMF IBSFN QMFBE OPI RHONDA 105 VDBMN DQDGS LNQEM ZLZRZ RNGVX ZALNA TERZV CZDGD MZQHZ GENKK KONSC DJHKC KKZAD RZAXZ SNMRH GBHGV RZVDG XZRNS XZKOS ZCNNF SHFMH BOMBAY 50 QFNWX YQFNW YSAQX FYNWY XQFNW SXYQF FXNYS AXYQF NASXY QFNAX HEAVEN 20 GOVGG OQNVU PKWQN UKWUK LOUIS	1 6 25 6 Crib is not encrypted. 2 6

Problem H. A Scheduling Problem

Time limit 1000 ms
Mem limit 65536 kB

There is a set of jobs, say $x_1, x_2, \ldots, x_n$, to be scheduled. Each job needs one day to complete. Your task is to schedule the jobs so that they can be finished in a minimum number of days. There are two types of constraints: *Conflict constraints* and *Precedence constraints*.

Conflict constraints: Some pairs of jobs cannot be done on the same day. (Maybe job x_i and job x_j need to use the same machine. So they must be done in different dates).

Precedence constraints: For some pairs of jobs, one needs to be completed before the other can start. For example, maybe job x_i cannot be started before job x_j is completed.

The scheduling needs to satisfy all the constraints.

To record the constraints, we build a graph G whose vertices are the jobs: $x_1, x_2, \ldots, x_n$. Connect x_i and x_j by an undirected edge if x_i and x_j cannot be done on the same day. Connect x_i and x_j by a directed edge from x_i to x_j if x_i needs to be completed before x_j starts.

If the graph is complicated, the scheduling problem is very hard. Now we assume that for our problems, the constraints are not very complicated: The graph G we need to consider are always trees (after omitting the directions of the edges). Your task is to find out the number of days needed in an optimal scheduling for such inputs. You can use the following result:

If G is a tree, then the number of days needed is either k or $k + 1$, where k is the maximum number of vertices contained in a directed path of G, i.e., a path $P = (x_1, x_2, \ldots, x_k)$, where for each $i = 1, 2, \ldots, k - 1$, there is a directed edge from x_i to x_{i+1}.

Figure 1 below is such an example. There are six jobs: 1, 2, 3, 4, 5, 6. From this figure, we know that job 1 and job 2 must be done in different dates. Job 1 needs to be done before job 3, job 3 before job 5, job 2 before job 4 and job 4 before job 6. It is easy to verify that the minimum days to finish all the jobs is 4 days. In this example, the maximum number k of vertices contained in a directed path is 3.

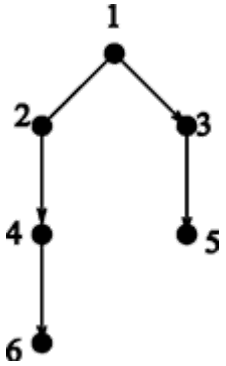


Figure 1: Example

Input

The input consists of a number of trees (whose edges may be directed or undirected), say T_1, T_2, …, T_m, where $m \leq 20$. Each tree has at most 200 vertices. We represent each tree as a rooted tree (just for convenience of presentation, the root is an arbitrarily chosen vertex). Information of each of the trees are contained in a number of lines. Each line starts with a vertex (which is a positive integer) followed by all its sons (which are also positive integers), then followed by a 0. Note that 0 is not a vertex and it indicates the end of that line. Now some of the edges are directed. The direction of an edge can be from father to son, and can also be from son to father. If the edge is from father to son, then we put a letter "d" after that son (meaning that it is a downward edge). If the edge is from son to father, then we put a letter "u" after that son (meaning that it is an upward edge). If the edge is undirected then we do not put any letter after the son.

The first case of the sample input below is the example in Figure 1.

Consecutive vertices (numbers or numbers with a letter after it) in a line are separated by a single space. A line containing a single 0 means the end of that tree. The next tree starts in the next line. Two consecutive lines of single 0 means the end of the input.

Output

The output contains one line for each test case. Each line contains a number, which is the minimum number of days to finish all the jobs in that test case.

Sample

Input	Output
1 2 3d 0 2 4d 0 3 5d 0 4 6d 0 0 1 2d 3u 4 0 0 1 2d 3 0 2 4d 5d 10 0 3 6d 7d 11 0 6 8d 9 12 0 0 1 2 3 4 0 2 5d 0 3 6d 0 4 7d 0 5 8d 0 6 9d 0 7 10d 0 0 0	4 3 4 3