Problem A: Knights of the Round Table

Introduction

Being a knight is a very attractive career: searching for the Holy Grail, saving damsels in distress, and drinking with the other knights are fun things to do. Therefore, it is not very surprising that in recent years the kingdom of King Arthur has experienced an unprecedented increase in the number of knights. There are so many knights now, that it is very rare that every Knight of the Round Table can come at the same time to Camelot and sit around the round table; usually only a small group of the knights is there, while the rest are busy doing heroic deeds around the country.

Knights can easily get over-excited during discussions—especially after a couple of drinks. After some unfortunate accidents, King Arthur asked the famous wizard Merlin to make sure that in the future no fights break out between the knights. After studying the problem carefully, Merlin realized that the fights can only be prevented if the knights are seated according to the following two rules:

- The knights should be seated such that two knights who hate each other should not be neighbors at the table. (Merlin has a list that says who hates whom.) The knights are sitting around a round table, thus every knight has exactly two neighbors.
- An odd number of knights should sit around the table. This ensures that if the knights cannot agree on something, then they can settle the issue by voting. (If the number of knights is even, then it can happen that "yes" and "no" have the same number of votes, and the argument goes on.)

Merlin will let the knights sit down only if these two rules are satisfied, otherwise he cancels the meeting. (If only one knight shows up, then the meeting is canceled as well, as one person cannot sit around a table.) Merlin realized that this means that there can be knights who cannot be part of any seating arrangements that respect these rules, and these knights will never be able to sit at the Round Table (one such case is if a knight hates every other knight, but there are many other possible reasons). If a knight cannot sit at the Round Table, then he cannot be a member of the Knights of the Round Table and must be expelled from the order. These knights have to be transferred to a less-prestigious order, such as the Knights of the Square Table, the Knights of the Octagonal Table, or the Knights of the Banana-Shaped Table. To help Merlin, you have to write a program that will determine the number of knights that must be expelled.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers $1 \leq n \leq 1000$ and $1 \leq m \leq 1000000$. The number n is the number of knights. The next m lines describe which knight hates which knight. Each of these m lines contains two integers k_1 and k_2, which means that knight number k_1 and knight number k_2 hate each other (the numbers k_1 and k_2 are between 1 and n).

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

Output

For each test case you have to output a single integer on a separate line: the number of knights that have to be expelled.

Sample Input

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

Sample Output

```
2
```

Problem B: The Cow Doctor

Introduction

Texas is the state having the largest number of cows in the US: according to the 2005 report of the National Agricultural Statistics Service, the bovine population of Texas is 13.8 million. This is higher than the population of the two runner-up states combined: there are only 6.65 million cows in Kansas and 6.35 millions cows in Nebraska.

There are several diseases that can threaten a herd of cows, the most feared being "Mad Cow Disease" or Bovine Spongiform Encephalopathy (BSE); therefore, it is very important to be able to diagnose certain illnesses. Fortunately, there are many tests available that can be used to detect these diseases.

A test is performed as follows. First a blood sample is taken from the cow, then the sample is mixed with a test material. Each test material detects a certain number of diseases. If the test material is mixed with a blood sample having any of these diseases, then a reaction takes place that is easy to observe. However, if a test material can detect several diseases, then we have no way to decide which of these diseases is present in the blood sample as all of them produce the same reaction. There are materials that detect many diseases (such tests can be used to rule out several diseases at once) and there are tests that detect only a few diseases (they can be used to make an accurate diagnosis of the problem).

The test materials can be mixed to create new tests. If we have a test material that detects diseases A and B; and there is another test material that detects diseases B and C, then they can be mixed to obtain a test that detects diseases A, B, and C. This means that if we have these two test materials, then there is no need for a test material that tests diseases A, B, and C—such a material can be obtained by mixing these two.

Producing, distributing, and storing many different types of test materials is very expensive, and in most cases, unnecessary. Your task is to eliminate as many unnecessary test materials as possible. It has to be done in such a way that if a test material is eliminated, then it should be possible to mix an equivalent test from the remaining materials. ("Equivalent" means that the mix tests exactly the same diseases as the eliminated material, not more, not less).

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 300$ of diseases, and the number $1 \leq m \leq 200$ of test materials. The next m lines correspond to the m test materials. Each line begins with an integer, the number $1 \leq k \leq 300$ of diseases that the material can detect. This is followed by k integers describing the k diseases. These integers are between 1 and n.

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

Output

For each test case, you have to output a line containing a single integer: the maximum number of test materials that can be eliminated.

Sample Input

```
10 5
2 1 2
2 2 3
3 1 2 3
4 1 2 3 4
1 4
3 7
1 1
1 2
1 3
2 1 2
2 1 3
2 3 2
3 1 2 3
0 0
```

Sample Output

```
2
4
```

Problem C: Wild West

Introduction

Once upon a time in the west... The quiet life of the villages on the western frontier are often stirred up by the appearance of mysterious strangers. A stranger might be a bounty hunter looking for a notorious villain, or he might be a dangerous criminal escaping the hand of justice. The number of strangers has become so large that they formed the Mysterious Strangers' Union. If you want to be a mysterious stranger, then you have to apply to the Union, and you have to pass three exams that test the three most important skills: shooting, fist-fighting, and harmonica playing. For each skill, the Admission Committee gives a score between 1 (worst) and m (best). Interestingly enough, there are no two members in the Union having exactly the same skills: for every two member, there is always at least one skill for which they have different scores. Furthermore, it turns out that for every possible combination of scores there is exactly one member having these scores. This means that there are exactly m^3 strangers in the union.

Recently, some members left the Union and they formed the Society of Evil Mysterious Strangers. The aim of this group is to commit as many evil crimes as possible, and they are quite successful at it. Therefore, the Steering Committee of the Union decided that a Hero is needed who will destroy this evil society. A Hero is a mysterious stranger who can defeat every member of the Society of Evil Mysterious Strangers. A Hero can defeat a member if the Hero has a higher score in *at least* one skill. For example, if the evil society has two members,

- Colonel Bill, with a score of 7 for shooting, 5 for knife throwing and 3 for harmonica playing, and
- Rabid Jack, with a score 10 for shooting, 6 for knife throwing and 8 for harmonica playing,

then a Hero with score 8 for shooting, 7 for knife throwing and 3 for harmonica playing can defeat both of them. However, someone with a score of 8 for shooting 6 for knife throwing and 8 for harmonica playing cannot be the Hero. Moreover, the Hero cannot be a member of the evil society.

Your task is to determine whether there is a member in the Union who can be the Hero. If so, then you have to count how many members are potential heroes.

Input

The input contains several blocks of test cases. Each block begins with a line containing two integers: the number $1 \leq n \leq 100000$ of members in the Society of Evil Mysterious Strangers and the maximum value $2 \leq m \leq 100000$ of the scores. The next n lines describe these members. Each line contains three integers between 1 and m: the scores for the three skills.

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

Output

For each test case, you have to output a single line containing the number of members in the Union who satisfy the requirements for becoming a Hero. If there is no such member, then output 0. It can be assumed that the output is always at most 10^{18}.

Sample Input

```
3 10
2 8 5
6 3 5
1 3 9
1 3
2 2 2
1 10000
2 2 2
0 0
```

Sample Output

```
848
19
999999999992
```

Problem D: Pixel Shuffle

Introduction

Shuffling the pixels in a bitmap image sometimes yields random looking images. However, it is not very difficult to show that by repeating the shuffling enough times, one finally recovers the original image (why?).

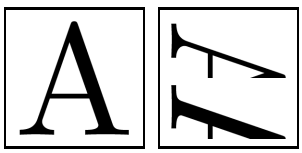
Your program should read a number n, and a series of elementary transformations that define a "shuffling" ϕ of $n \times n$ images. Then, your program should compute the minimal number m ($m > 0$), such that m applications of ϕ always yield the original $n \times n$ image.

For instance, if ϕ is counter-clockwise 90° rotation then $m = 4$.

Input

The input contains several blocks of test cases. Each case consists of two lines. The first line contains a single even integer $2 \leq n \leq 2^{10}$. The number n is the size of images, one image is represented internally by an $n \times n$ pixel matrix (a_i^j), where i is the row number and j is the column number. The pixel in the upper left corner is at row 0 and column 0.

The second line is a non-empty list of at most 32 words, separated by spaces. Valid words are the keywords **id**, **rot**, **sym**, **bhsym**, **bvsym**, **div** and **mix**, or a keyword followed by "-". Each keyword **key** designates an elementary transformation (as defined by Figure 1), and **key-** designates the inverse of transformation **key**. For instance, **rot-** is the inverse of a counter-clockwise 90° rotation, that is a clockwise 90° rotation. Finally, the list k_1, k_2, ..., k_p designates the compound transform $\phi = k_1 \circ k_2 \circ \cdots \circ k_p$. For instance, "**bvsym rot-**" is the transformation that first performs a clockwise 90° rotation and then vertical symmetry on the lower half of the image:



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

Output

Your program should output a single line whose contents is the minimal number m ($m > 0$) such that ϕ^m is the identity. You may assume that for all test input you have $m < 2^{31}$.

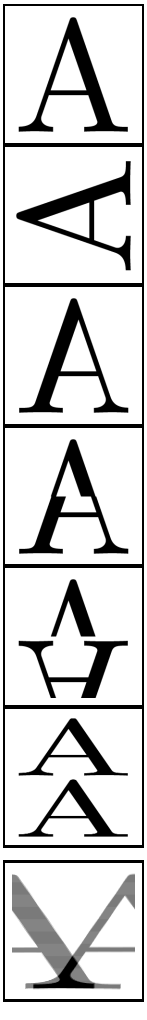
Sample Input

```
256
rot- div rot div
256
bvsym div mix
0
```

Sample Output

```
8
63457
```

Figure 1: Transformations of image (a_i^j) into image (b_i^j)



id, identity. Nothing changes : $b_i^j = a_i^j$.

rot, counter-clockwise 90° rotation

sym, horizontal symmetry : $b_i^j = a_i^{n-1-j}$

bhsym, horizontal symmetry applied to the lower half of image : when $i \geq n/2$, then $b_i^j = a_i^{n-1-j}$. Otherwise $b_i^j = a_i^j$.

bvsym, vertical symmetry applied to the lower half of image ($i \geq n/2$)

div, division. Rows 0, 2, ..., $n-2$ become rows 0, 1, ... $n/2-1$, while rows 1, 3, ... $n-1$ become rows $n/2$, $n/2+1$, ... $n-1$.

mix, row mix. Rows $2k$ and $2k+1$ are interleaved. The pixels of row $2k$ in the new image are $a_{2k}^0, a_{2k+1}^0, a_{2k}^1, a_{2k+1}^1, \ldots a_{2k}^{n/2-1}, a_{2k+1}^{n/2-1}$, while the pixels of row $2k+1$ in the new image are $a_{2k}^{n/2}, a_{2k+1}^{n/2}, a_{2k}^{n/2+1}, a_{2k+1}^{n/2+1}, \ldots a_{2k}^{n-1}, a_{2k+1}^{n-1}$

Problem E: Find the Clones

Introduction

Doubleville, a small town in Texas, was attacked by the aliens. They have abducted some of the residents and taken them to the a spaceship orbiting around earth. After some (quite unpleasant) human experiments, the aliens cloned the victims, and released multiple copies of them back in Doubleville. So now it might happen that there are 6 identical person named Hugh F. Bumblebee: the original person and its 5 copies. The Federal Bureau of Unauthorized Cloning (FBUC) charged you with the task of determining how many copies were made from each person. To help you in your task, FBUC have collected a DNA sample from each person. All copies of the same person have the same DNA sequence, and different people have different sequences (we know that there are no identical twins in the town, this is not an issue).

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 20000$ people, and the length $1 \leq m \leq 20$ of the DNA sequences. The next n lines contain the DNA sequences: each line contains a sequence of m characters, where each character is either `'A'`, `'C'`, `'G'` or `'T'`.

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

Output

For each test case, you have to output n lines, each line containing a single integer. The first line contains the number of different people that were not copied. The second line contains the number of people that were copied only once (i.e., there are two identical copies for each such person.) The third line contains the number of people that are present in three identical copies, and so on: the i-th line contains the number of persons that are present in i identical copies. For example, if there are 11 samples, one of them is from John Smith, and all the others are from copies of Joe Foobar, then you have to print 1 in the first and the tenth lines, and 0 in all the other lines.

Sample Input

```
9 6
AAAAAA
ACACAC
GTTTTG
ACACAC
GTTTTG
ACACAC
ACACAC
TCCCCC
TCCCCC
0 0
```

Sample Output

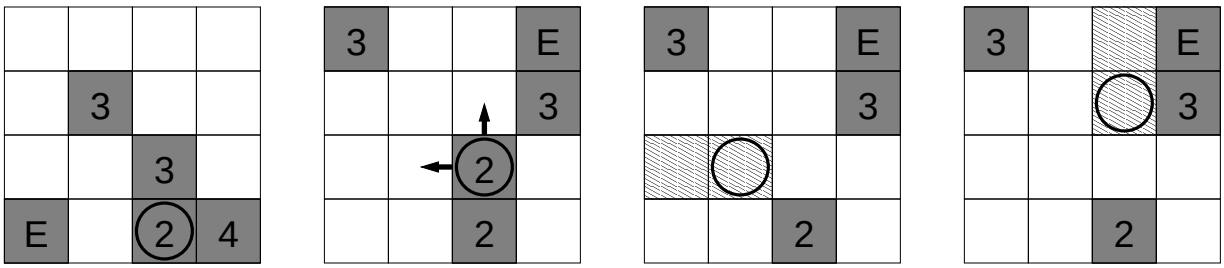
```
1
2
0
1
0
0
0
0
0
```

Problem F: The Warehouse

Introduction

Secret Agent Θ-7 has found the secret weapon warehouse of the mad scientist Dr. Matroid. The warehouse is full of large boxes (possibly with deadly weapons inside the boxes). While inspecting the warehouse, Θ-7 accidentally triggered the alarm system. The warehouse has a very effective protection against intruders: if the alarm is triggered, then the floor is filled with deadly acid. Therefore, the only way Θ-7 can escape is to climb onto the boxes and somehow reach the exit on top of them. The exit is a hole in the ceiling, if Θ-7 can climb through this hole then he can escape using the helicopter parked on the roof. There is a ladder and a box below the hole, thus the goal is to reach this box.

The floor of the warehouse can be divided into a grid containing $n \times n$ cells, the size of each cell is 1 meter $\times$ 1 meter. Each cell is either fully occupied by one box or unoccupied. Each box is rectangular: the size of the base is 1 meter $\times$ 1 meter, and the height is either 2, 3, or 4 meters. In figure (a), you can see an example warehouse, where the numbers show the height of the boxes, E shows the exit, and the circle shows that Secret Agent Θ-7 is currently on the top of that box.



Θ-7 can do two things:

If he is standing on top of a box, and in an adjacent cell there is another box, then he can move to the top of this other box. For example, in the situation depicted in figure (a), he can move either to north or east, but not to west or south. Note that only these four directions are allowed, diagonal moves are not possible. The height difference between the two boxes does not matter.

The second thing Θ-7 can do is that he can topple the box he is standing on in one of the four directions. The effect of toppling is best show by an example: in the situation shown in figure (b), he can topple the box west (figure (c)) or north (figure (d)). If a box of height h is toppled north (west, south, etc.) then it will occupy h consecutive cells to the north (west, south, etc.) of its original position. The original position will be unoccupied (but can be later occupied again by toppling another box). A box can only be toppled if the cells where it will fall are unoccupied. For example, in figure (a), the box where Θ-7 is standing cannot be toppled in any of the four directions.

By toppling a box, Θ-7 jumps one step in the direction that the box is toppled (see figures (c) and (d)). If a box is toppled, then it cannot be toppled again later. Recall that there is a box below the exit (at the cell marked with E in the figure), thus it is not possible to topple a box over this cell.

The alarm system will soon release mutant poisonous biting bats, so Θ-7 has to leave the warehouse as quickly as possible. You have to help him by writing a program that will determine the minimum number of steps required to reach the exit. Moving to an adjacent box, or toppling a box is counted as one step.

Input

The input contains several blocks of test cases. The first line of each block contains three integers: the size $1 \leq n \leq 8$ of the warehouse, and two integers i, j that describe the starting position of the secret agent. These numbers are between 1 and n; the row number is given by i, the column number is given by

j. The next n lines describe the warehouse. Each line contains a string of n characters. Each character corresponds to a cell of the warehouse. If the character is `'.'`, then the cell is unoccupied. The characters `'2'`, `'3'` and `'4'` correspond to boxes of height 2, 3 and 4, respectively. Finally, the character `'E'` shows the location of the exit.

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

Output

For each test case, you have to output a single line containing an integer: the minimum number of steps required to reach the exit. If it is not possible to reach the exit, then output the text `'Impossible.'` (without quotes).

Sample Input

```
5 5 3
.2..E
...2.
4....
....4
..2..
0 0 0
```

Sample Output

```
18
```

Problem G: Widget Factory

Introduction

The widget factory produces several different kinds of widgets. Each widget is carefully built by a skilled widgeteer. The time required to build a widget depends on its type: the simple widgets need only 3 days, but the most complex ones may need as many as 9 days.

The factory is currently in a state of complete chaos: recently, the factory has been bought by a new owner, and the new director has fired almost everyone. The new staff know almost nothing about building widgets, and it seems that no one remembers how many days are required to build each different type of widget. This is very embarrassing when a client orders widgets and the factory cannot tell the client how many days are needed to produce the required goods. Fortunately, there are records that say for each widgeteer the date when he started working at the factory, the date when he was fired and what types of widgets he built. The problem is that the record does not say the exact date of starting and leaving the job, only the day of the week. Nevertheless, even this information might be helpful in certain cases: for example, if a widgeteer started working on a Tuesday, built a Type 41 widget, and was fired on a Friday, then we know that it takes 4 days to build a Type 41 widget. Your task is to figure out from these records (if possible) the number of days that are required to build the different types of widgets.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 300$ of the different types, and the number $1 \leq m \leq 300$ of the records. This line is followed by a description of the m records. Each record is described by two lines. The first line contains the total number $1 \leq k \leq 10000$ of widgets built by this widgeteer, followed by the day of week when he/she started working and the day of the week he/she was fired. The days of the week are given by the strings `'MON'`, `'TUE'`, `'WED'`, `'THU'`, `'FRI'`, `'SAT'` and `'SUN'`. The second line contains k integers separated by spaces. These numbers are between 1 and n, and they describe the different types of widgets that the widgeteer built. For example, the following two lines mean that the widgeteer started working on a Wednesday, built a Type 13 widget, a Type 18 widget, a Type 1 widget, again a Type 13 widget, and was fired on a Sunday.

```
4 WED SUN
13 18 1 13
```

Note that the widgeteers work 7 days a week, and they were working on every day between their first and last day at the factory (if you like weekends and holidays, then do not become a widgeteer!).

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

Output

For each test case, you have to output a single line containing n integers separated by spaces: the number of days required to build the different types of widgets. There should be no space before the first number or after the last number, and there should be exactly one space between two numbers. If there is more than one possible solution for the problem, then write `'Multiple solutions.'` (without the quotes). If you are sure that there is no solution consistent with the input, then write `'Inconsistent data.'` (without the quotes).

Sample Input

```
2 3
2 MON THU
1 2
3 MON FRI
1 1 2
3 MON SUN
1 2 2
10 2
1 MON TUE
3
1 MON WED
3
0 0
```

Sample Output

```
8 3
Inconsistent data.
```

Problem H: Martian Mining

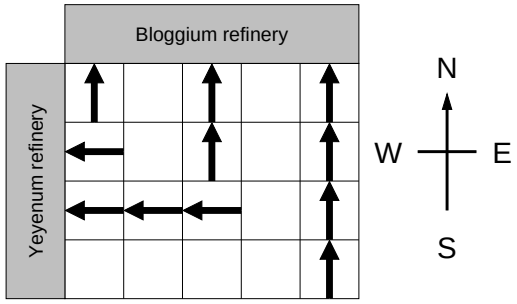
Introduction

The NASA Space Center, Houston, is less than 200 miles from San Antonio, Texas (the site of the ACM Finals this year). This is the place where the astronauts are trained for Mission Seven Dwarfs, the next giant leap in space exploration. The Mars Odyssey program revealed that the surface of Mars is very rich in yeyenum and bloggium. These minerals are important ingredients for certain revolutionary new medicines, but they are extremely rare on Earth. The aim of Mission Seven Dwarfs is to mine these minerals on Mars and bring them back to Earth.

The Mars Odyssey orbiter identified a rectangular area on the surface of Mars that is rich in minerals. The area is divided into cells that form a matrix of n rows and m columns, where the rows go from east to west and the columns go from north to south. The orbiter determined the amount of yeyenum and bloggium in each cell. The astronauts will build a yeyenum refinement factory west of the rectangular area and a bloggium factory to the north. Your task is to design the conveyor belt system that will allow them to mine the largest amount of minerals.

There are two types of conveyor belts: the first moves minerals from east to west, the second moves minerals from south to north. In each cell you can build either type of conveyor belt, but you cannot build both of them in the same cell. If two conveyor belts of the same type are next to each other, then they can be connected. For example, the bloggium mined at a cell can be transported to the bloggium refinement factory via a series of south-north conveyor belts.

The minerals are very unstable, thus they have to be brought to the factories on a straight path without any turns. This means that if there is a south-north conveyor belt in a cell, but the cell north of it contains an east-west conveyor belt, then any mineral transported on the south-north conveyor belt will be lost. The minerals mined in a particular cell have to be put on a conveyor belt immediately, in the same cell (thus they cannot start the transportation in an adjacent cell). Furthermore, any bloggium transported to the yeyenum refinement factory will be lost, and vice versa.



Your program has to design a conveyor belt system that maximizes the total amount of minerals mined, i.e., the sum of the amount of yeyenum transported to the yeyenum refinery and the amount of bloggium transported to the bloggium refinery.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 500$ of rows, and the number $1 \leq m \leq 500$ of columns. The next n lines describe the amount of yeyenum that can be found in the cells. Each of these n lines contains m integers. The first line corresponds to the northernmost row; the first integer of each line corresponds to the westernmost cell of the row. The integers are between 0 and 1000. The next n lines describe in a similar fashion the amount of bloggium found in the cells.

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

Output

For each test case, you have to output a single integer on a separate line: the maximum amount of minerals that can be mined.

Sample Input

```
4 4
0 0 10 9
1 3 10 0
4 2 1 3
1 1 20 0
10 0 0 0
1 1 1 30
0 0 5 5
5 10 10 10
0 0
```

Sample Output

```
98
```

Problem I: Word Rings

Introduction

A word ring is a sequence of words where the last two letters of each word are the same as the first two letters of the next word (and the last two letters of the last word are the same as the first two letters of the first word). For example, the following sequence is a word ring:

```
intercommunicational
alkylbenzenesulfonate
tetraiodophenolphthalein
```

Your task is to write a program that, given a list of words, finds a word ring. You have to make the word ring as impressive as possible: the *average* length of the words in the ring has to be as large as possible. In the above example, the average length is $(20+21+24)/3 \approx 21.6666$, which makes it somewhat impressive. Note that each word can be used at most once in the ring, and the ring can consist of a single word.

Input

The input contains several blocks of test cases. Each case begins with a line containing a single integer $1 \leq n \leq 100000$, the number of possible words that can be used. The next n lines contain these words. The words contain only the characters `'a'–'z'` and the length of each word is at most 1000.

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

Output

For each test case in the input, you have to output a single number on a separate line: the maximum average length of a ring composed from (a subset of) the words given in the input. The average length should be presented as a real number with two digits of precision. If it is not possible to compose a ring from these words, then output `'No solution.'` (without quotes). To avoid rounding problems, we accept solutions with a maximum of ± 0.01 error.

Sample Input

```
3
intercommunicational
alkylbenzenesulfonate
tetraiodophenolphthalein
0
```

Sample Output

```
21.66
```

Problem J: Nuclear Plants

Introduction

The Great Plain of Algorithmia plays an extremely important role in the agriculture of the Bandulu Kingdom: this is the only place where barley (*Hordeum vulgare*), an essential ingredient of beer, can be produced. Unfortunately, it is not possible to grow barley on the full area of the plain, as several nuclear plants have recently been built, and you cannot grow barley near a nuclear plant (since you do not want to produce giant-size, aggressive, man-eating barley-mutants). Your task is to write a program that determines the size of the area that can be used for growing barley.

The Great Plain of Algorithmia is an n km $\times$ m km rectangle, the coordinates of the four corners being $(0,0)$, $(0,m)$, $(n,0)$ and (n,m). There are two types of nuclear plants: small and large. You are not allowed to grow barley within 0.58km of a small nuclear plant or within 1.31km of a large nuclear plant.

Input

The input contains several blocks of test cases. Each block begins with a line containing four integers: $1 \leq n, m \leq 10000$ describe the size of the plain, $k_s \leq 100$ is the number of small nuclear plants, and $k_\ell \leq 100$ is the number of large nuclear plants. The next k_s lines describe the coordinates of the small nuclear plants, each line contains two integers $0 \leq x \leq n$ and $0 \leq y \leq m$. The next k_ℓ lines describe the large nuclear plants in a similar fashion.

The input is terminated by a block with $n = m = k_s = k_\ell = 0$.

Output

For each test case, you have to output a single line containing the area that can be used for growing barley. This number should be a real value with two digits of precision. To avoid rounding problems, we accept solutions with a maximum of ± 0.01 error.

Sample Input

```
10 10 2 2
2 2
4 4
5 6
1 8
10 10 1 0
5 5
0 0 0 0
```

Sample Output

```
87.46
98.94
```