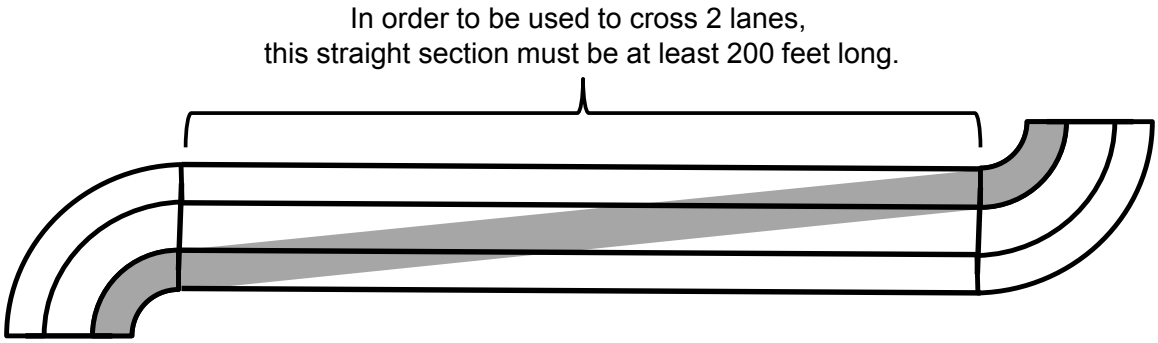
A: Sunday Drive

After wracking your brains at a programing contest on Saturday, you'd like to relax by taking a leisurely Sunday drive. But, gasoline is so expensive nowadays! Maybe, by creatively changing lanes, you can minimize the distance you travel and save some money!

You will be given a description of several sections of a highway. All sections will have the same number of lanes. Think of your car as a point mass, moving down the center of the lane. Each lane will be 10 feet wide. There are two kinds of highway sections: curved and straight. You can only change lanes on straight sections, and it takes a minimum of 100 feet of the straight section to move over one lane. You can take longer than that, of course, if you choose.

All curve sections will make 90 degree turns. You cannot change lanes on a curve section. In addition, you must be driving along the exact middle of a lane during a turn. So during a turn your position will be 5 feet, or 15 feet, or 25 feet from the edge, etc.

Given a description of a highway, compute the minimum total distance required travel the entire highway, including curves and lane changes. You can start, and end, in any lane you choose. Assume that your car is a point mass in the center of the lane. The highway may cross over/under itself, but the changes in elevation are miniscule, so you shouldn't worry about their impact on your distance traveled.



Input

There will be several test cases in the input. Each test case will begin with two integers

N M

Where **N** ($1 \le \mathbf{N} \le 1{,}000$) is the number of segments, and **M** ($2 \le \mathbf{M} \le 10$) is the number of lanes.

On each of the next **N** lines will be a description of a segment, consisting of a letter and a number, with a single space between them:

T K

The letter **T** is one of `S`, `L`, or `R` (always capital). This indicates the type of the section: a straight section (`S`), a left curve (`L`) or a right curve (`R`). If the section is a straight section, then the number **K** ($10 \le K \le 10{,}000$) is simply its length, in feet. If the section is a right or left curve, then the number **K** ($10 \le K \le 10{,}000$) is the radius of the inside edge of the highway, again in feet. There will never be consecutive straight sections in the input, but multiple consecutive turns are possible. The input will end with a line with two 0s.

Output

For each test case, print a single number on its own line, indicating the minimum distance (in feet) required to drive the entire highway. The number should be printed with exactly two decimal places, rounded. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
3 3
R 100
S 1000
L 100
9 5
S 2500
L 500
S 2000
L 500
S 5000
L 500
S 2000
L 500
S 2500
5 4
L 100
L 100
L 100
L 100
L 100
0 0
```

Sample Output

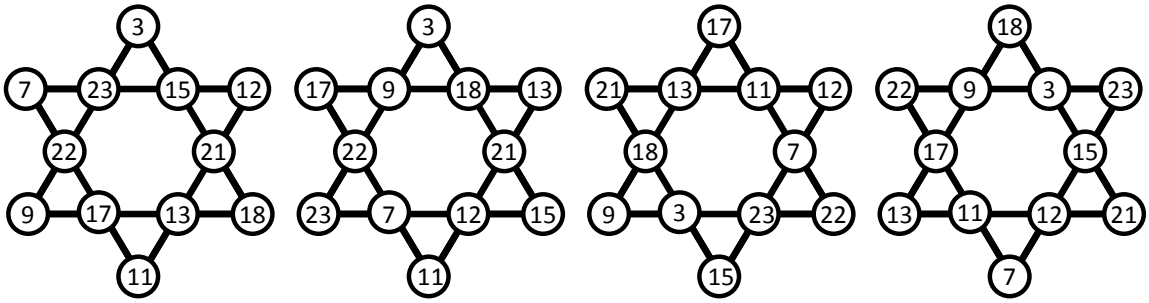
```
1330.07
17173.01
824.67
```

B: Hexagram

A Hexagram is a 6-pointed star, sometimes called the Star of David. Given these numbers:

```
3 17 15 18 11 22 12 23 21 7 9 13
```

There are four unique ways of assigning the numbers to vertices of the hexagram such that all of the sets of four numbers along the lines have the same sum (57 in this case). All other ways may be obtained from these by rotation and/or reflection.



Given 12 distinct numbers, in how many ways, disregarding rotations and reflections, can you assign the numbers to the vertices such that the sum of the numbers along each of 6 straight lines passing through 4 vertices is the same?

The Input

There will be several test cases in the input. Each test case will consist of twelve unique positive integers on a single line, with single spaces separating them. All of the numbers will be less than 1,000,000. The input will end with a line with twelve 0s.

The Output

For each test case, output the number of ways the numbers can be assigned to vertices such that the sum along each line of the hexagram is the same. Put each answer on its own line. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
3 17 15 18 11 22 12 23 21 7 9 13
1 2 3 4 5 6 7 8 9 10 11 13
0 0 0 0 0 0 0 0 0 0 0 0
```

Sample Output

```
4
0
```

acm International Collegiate Programming Contest

IBM event sponsor

C: Flooring Tiles

You want to decorate your floor with square tiles. You like rectangles. With six square flooring tiles, you can form exactly two unique rectangles that use all of the tiles: 1x6, and 2x3 (6x1 is considered the same as 1x6. Likewise, 3x2 is the same as 2x3). You can also form exactly two unique rectangles with four square tiles, using all of the tiles: 1x4, and 2x2.

Given an integer **N**, what is the smallest number of square tiles needed to be able to make exactly **N** unique rectangles, and no more, using all of the tiles? If **N**=2, the answer is 4.

Input

There will be several test cases in the input. Each test case will consist of a single line containing a single integer **N** ($1 \le N \le 75$), which represents the number of desired rectangles. The input will end with a line with a single 0.

Output

For each test case, output a single integer on its own line, representing the smallest number of square flooring tiles needed to be able to form exactly **N** rectangles, and no more, using all of the tiles. The answer is guaranteed to be at most 10^{18}. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
2
16
19
0
```

Sample Output

```
4
840
786432
```

acm International Collegiate Programming Contest

IBM event sponsor

D: Vive la Difference!

Take any four positive integers: **a**, **b**, **c**, **d**. Form four more, like this:

|a-b| |b-c| |c-d| |d-a|

That is, take the absolute value of the differences of **a** with **b**, **b** with **c**, **c** with **d**, and **d** with **a**. (Note that a zero could crop up, but they'll all still be non-negative.) Then, do it again with these four new numbers. And then again. And again. Eventually, all four integers will be the same. For example, start with `1,3,5,9`:

```
1 3 5 9
2 2 4 8 (1)
0 2 4 6 (2)
2 2 2 6 (3)
0 0 4 4 (4)
0 4 0 4 (5)
4 4 4 4 (6)
```

In this case, the sequence converged in 6 steps. It turns out that in all cases, the sequence converges very quickly. In fact, it can be shown that if all four integers are less than 2^n, then it will take no more than 3*n steps to converge!

Given **a**, **b**, **c** and **d**, figure out just how quickly the sequence converges.

Input

There will be several test cases in the input. Each test case consists of four positive integers on a single line (1 ≤ **a**,**b**,**c**,**d** ≤ 2,000,000,000), with single spaces for separation. The input will end with a line with four 0s.

Output

For each test case, output a single integer on its own line, indicating the number of steps until convergence. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

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

Sample Output

```
6
4
0
```

acm International Collegiate Programming Contest

IBM event sponsor

E: Robot Navigation

A robot has been sent to explore a remote planet. To specify a path the robot should take, a program is sent each day. The program consists of a sequence of the following commands:

- `FORWARD` *X*: move forward by *X* units.
- `TURN LEFT`: turn left (in place) by 90 degrees.
- `TURN RIGHT`: turn right (in place) by 90 degrees.

The robot also has sensor units which allow it to obtain a map of its surrounding area. The map is represented as a grid. Some grid points contain hazards (e.g. craters) and the program must avoid these points or risk losing the robot.

Naturally, if the initial location of the robot, the direction it is facing, and its destination position are known, it is best to send the shortest program (one consisting of the fewest commands) to move the robot to its destination (we do not care which direction it faces at the destination). You are more interested in knowing the number of different shortest programs that can move the robot to its destination. However, the number of shortest programs can be very large, so you are satisfied to compute the number as a remainder modulo 1,000,000.

Input

There will be several test cases in the input. Each test case will begin with a line with two integers

N M

Where **N** is the number of rows in the grid, and **M** is the number of columns in the grid ($2 \le$ **N, M** ≤ 100). The next **N** lines of input will have **M** characters each. The characters will be one of the following:

`'.'` Indicating a navigable grid point.

`'*'` Indicating a crater (i.e. a non-navigable grid point).

`'X'` Indicating the target grid point. There will be exactly one `'X'`.

`'N'`, `'E'`, `'S'`, or `'W'` Indicating the starting point and initial heading of the robot. There will be exactly one of these. Note that the directions mirror compass directions on a map: `N` is North (toward the top of the grid), `E` is East (toward the right of the grid), `S` is South (toward the bottom of the grid) and `W` is West (toward the left of the grid).

There will be no spaces and no other characters in the description of the map. The input will end with a line with two 0s.

Output

For each test case, output two integers on a single line, with a single space between them. The first is the length of a shortest possible program to navigate the robot from its starting point to the target, and the second is the number of different programs of that length which will get the robot to the target (modulo 1,000,000). If there is no path from the robot to the target, output two zeros separated by a single space. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
5 6
*....X
.....*
.....*
.....*
N....*
6 5
....X
.****
.****
.****
.****
N****
3 3
.E.
***
.X.
0 0
```

Sample Output

```
6 4
3 1
0 0
```

acm International Collegiate Programming Contest

IBM event sponsor

F: Vampire Numbers

The number `1827` is an interesting number, because `1827=21*87`, and all of the same digits appear on both sides of the '='. The number `136948` has the same property: `136948=146*938`.

Such numbers are called *Vampire Numbers*. More precisely, a number **v** is a Vampire Number if it has a pair of factors, **a** and **b**, where **a*b=v**, and together, **a** and **b** have exactly the same digits, in exactly the same quantities, as **v**. None of the numbers **v**, **a** or **b** can have leading zeros. The mathematical definition says that **v** should have an even number of digits and that **a** and **b** should have the same number of digits, but for the purposes of this problem, we'll relax that requirement, and allow **a** and **b** to have differing numbers of digits, and **v** to have any number of digits. Here are some more examples:

```
126    = 6  * 21
10251  = 51 * 201
702189 = 9  * 78021
29632  = 32 * 926
```

Given a number **X**, find the smallest Vampire Number which is greater than or equal to **X**.

Input

There will be several test cases in the input. Each test case will consist of a single line containing a single integer **X** (10 ≤ **X** ≤ 1,000,000). The input will end with a line with a single 0.

Output

For each test case, output a single integer on its own line, which is the smallest Vampire Number which is greater than or equal to **X**. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
10
126
127
5000
0
```

Sample Output

```
126
126
153
6880
```