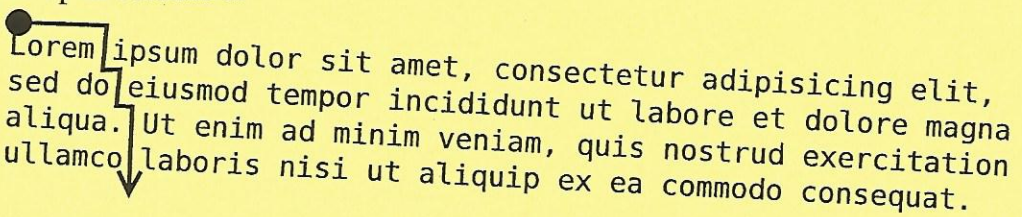
Problem A: Text Roll

Take some text. Put a small ball at the top of the first letter of the first word of the first sentence. The ball is drawn via gravity downwards. The text is also at a slight angle, so the ball wants to also move towards the right. The ball can freely move between the lines, and can drop through spaces. Considering the first column to be column 1, what column will the ball end up in? In this example, the ball ends up in column 8.



Input

There will be multiple test cases in the input. Each test case will begin with an integer n ($1 \leq n \leq 1,000$) on its own line, indicating the number of lines of text. On each of the next n lines will be text, consisting of printable ASCII characters and spaces. There will be no tabs, nor any other unprintable characters. Each line will be between 1 and 100 characters long.

The input will end with a line containing a single 0.

Output

For each test case, output a single integer on its own line, indicating the column from which the ball will drop.

Example

Given the input

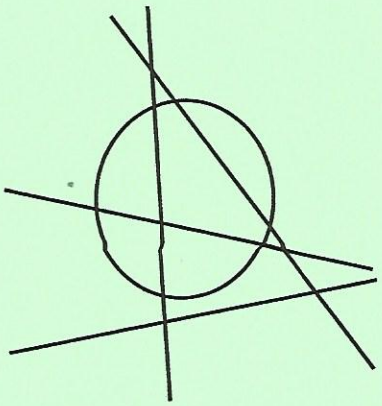
```
4
Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
3
Supercalifragilisticexpialidocious! Can you handle
short
lines?
0
```

the output should be

```
8
36
```

Problem B: Cut the Cake

Given a circle, and a list of lines, how many parts has the circle been cut into? In the example shown here, four lines cut the circle into five parts.



Input

There will be multiple test cases in the input. Each test case will begin with four integers, r ($1 \leq r \leq 1,000$), x, y ($-1,000 \leq x, y \leq 1,000$), and n ($0 \leq n \leq 1,000$), where r is the radius of a circle, x and y are the coordinates of the center of the circle, and n is the number of lines. On each of the next n lines will be four integers, $x1$, $y1$, $x2$, and $y2$ ($-1,000 \leq x1, y1, x2, y2 \leq 1,000$). These describe a line passing through $(x1, y1)$ and $(x2, y2)$. Note that we are interested in the whole line, not just the segment between $(x1, y1)$ and $(x2, y2)$.

In any test case, no more than two lines will intersect at any point, and no lines will be tangent to the circle. No lines within any test case will be coincident.

End of input will be indicated by a line with four 0s.

Output

For each test case, output a single integer, indicating the number of parts that the circle is cut into.

Example

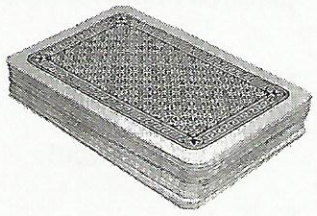
Given the input

```
1 0 0 2
0 0 0 1
-1 0 20 0
16 1 7 4
-15 -9 14 -11
-4 30 -3 -20
-20 12 -10 7
17 10 31 0
0 0 0 0
```

the output should be

```
4
5
```

Problem C: Split Decision



Alan has wandered into a casino featuring a new card game, *Split Decision*. On each hand of this game, each player antes[1] one chip, then the dealer receives two cards face up and each player receives a single card, face up. Each card has a value somewhere in the range from 2 to 11, inclusive. The values of the dealer's cards are added up, and a player must beat that total to win. (The dealer wins on ties.)

On his turn, Alan must decide to do one of the following

- "fold" his hand (losing the chip he has already paid)
- "stay", in which case he must ante up another chip to receive a second card from the deck. If he wins, Alan recovers his two-chip ante and wins a further two chips.
- "split", in which case he must ante up another two chips and receives one card to add to his original hand and another two cards that are added together as a separately scored hand. These cards are dealt in that order – Alan does not get to decide which card goes into which hand.

  On a split, Alan can win with both pairs, in which case he recovers his 3-chip ante and wins an additional 3 chips. He might lose with both pairs, in which case he loses the three chips he has anted. Finally, he might win with one pair and lose with the other, in which case his three chips are returned to him and he neither wins nor loses on that round.

Alan is very good at counting cards. As the game progresses, he can accurately keep track of what cards have already been played and, therefore, what cards remain in the deck for future play. Of course, he does not know in what order the remaining cards will appear.

Given a list of card values remaining in the deck, the values of the dealer's two cards, and the value of Alan's current card, determine whether Alan should fold, stay, or split in order to maximize his probable earnings (or minimize his probable loss) on this hand. If two of the options are tied for optimum win/loss, choose the option that risks the fewest total chips.

Input

The input may contain multiple test cases.

Each test case consists of two lines.

The first line contains the value of Alan's card and the values of the two dealer cards. Each card value is presented as an integer in the range 2 . . . 11, inclusive. A value of zero for the first of these integers signals end of input and does not constitute a valid test case.

The second line contains 10 integers in the range 0 . . . 4, indicating the number of cards remaining in the deck having values 2 . . . 11, respectively. The three cards from the first line are not included in these counts of remaining cards. The sum of the integers on this line will not exceed 49 nor fall below 3.

[1] To "ante" means to pay an amount with the possibility of winning it back.

Output

For each dataset, print a single line containing only the word "fold", "stay", or "split", left-justified and with no blank spaces, indicating the preferred action as outlined above.

Example

Given the input

```
5 2 10
1 2 3  1 2 1  1 0 0  0
5 2 10
0 0 1  0 2 1  1 4 4  4
11 2 10
1 2 3  1 2 1  1 0 0  0
0 0 0
```

the output should be

```
fold
split
stay
```

Problem D: Count Your Cousins

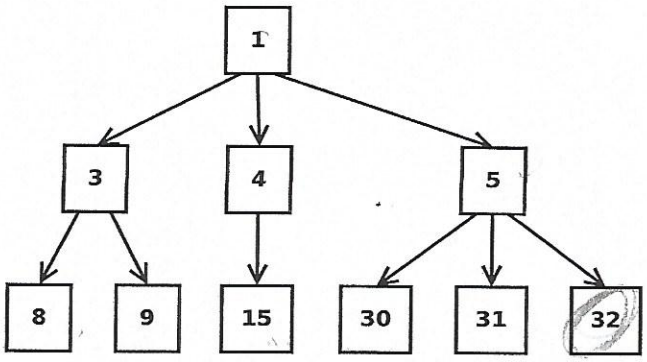
A tree is formed from a strictly increasing sequence of integers as follows:

- The first integer in the sequence is the root of the tree
- The next set of consecutive integers in the sequence describes the children of the root. The first of these will be greater than root+1.
- From there, each set of consecutive integers describes the children of the lowest numbered node which does not yet have children.

For example, the sequence:

```
1 3 4 5 8 9 15 30 31 32
```

would produce the family tree:



Two nodes are considered to be *Cousins* if they have different parents, but the same grandparent. Given a tree and a particular node of that tree, count the number of Cousins of the node.

Input

There will be multiple test cases in the input. Each test case will consist of two lines.

The first line will contain two integers, n ($1 \leq n \leq 1,000$) and k ($1 \leq k \leq 1,000,000$), where n is the number of nodes in the tree, and k is the particular node of interest. End of input will be indicated by a line with two 0s.

The second line will consist of n integers, all in the range $1 \ldots 1,000,000$ and guaranteed to be strictly increasing. These define the tree, in the manner described above.

The input k from the first line is guaranteed to be one of the integers on the second line.

Output

For each test case, output a single integer, indicating the number cousins of node k.

Example

Given the input

```
10 16
1 3 4 5 9 15 16 30 31 32
12 9
3 5 6 8 9 10 13 15 16 22 23 25
10 4
1 3 4 5 8 9 15 30 31 32
0 0
```

the output should be

```
4
1
0
```

Problem E: Ping!

Suppose you are tracking some satellites. Each satellite broadcasts a "ping" at a regular interval, and the intervals are unique (that is, no two satellites ping at the same interval). You need to know which satellites you can hear from your current position. The problem is that the pings cancel each other out. If an even number of satellites ping at a given time, you won't hear anything, and if an odd number ping at a given time, it sounds like a single satellite. All of the satellites ping at time 0, and then each pings regularly at its unique interval.

Given a sequence of pings and non-pings (silences), starting at time 0, which satellites can you determine that you can hear from where you are?

The sequence you're given may, or may not, be long enough to include all of the satellites' ping intervals. Just report about the ones with a short enough interval to be in the sequence of pings. There may be satellites that ping at time 0, but the sequence isn't long enough for you to hear their next ping. You don't have enough information to report about these satellites. Just report about the ones with an interval short enough to be in the sequence of pings.

Input

There will be multiple test cases in the input. Each test case will consist of a single string on its own line, with from 2 to 1,000 characters. The first character represents time 0, the next represents time 1, and so on. Each character will either be a 0 or a 1, with 1 indicating that a ping can be heard at that time and 0 indicating that no ping is heard. Each input is guaranteed to have at least one satellite that can be heard.

End of input will be indicated by a line with a single 0.

Output

For each test case, output a list of integers on a single line, indicating the intervals of the satellites that can be determined unambiguously from the input. Output the intervals in order from smallest to largest, with a single space between them.

Example

Given the input

```
01000101101000
1001000101001000
0
```

the output should be

```
1 2 3 6 8 10 11 13
3 6 7 12 14 15
```

Problem F: Star Simulation

In massive simulations of star systems, we don't want to have to model the gravitational effects of pairs of bodies that are too far away from each other, because that will take up excess computing power (and their effects on each other are negligible). We want to consider the effects two objects have on each other only if the Euclidean distance between the two is less than k.

Given a list of n points in space, how many have a distance of less than k apart from each other?

Input

There will be multiple test cases in the input.

Each test case will begin with a line with two integers, n ($2 \leq n \leq 100,000$) and k ($0 < k \leq 10^9$), where n is the number of points, and k is the desired maximum distance.

On each of the following n lines will be three integers x, y and z, ($-10^9 \leq x, y, z \leq 10^9$) which are the (x, y, z) coordinates of one point. Because stars systems are sparse and not uniformly distributed through space, it is guaranteed that no more than $100,000$ pairs of points will be within k of each other.

End of input will be indicated by a line with two 0s.

Output

For each test case, output a single integer indicating the number of unique pairs of points that are less than k apart from each other.

Example

Given the input

```
7 2
0 0 0
1 0 0
1 2 0
1 2 3
1000 1000 1000
1001 1001 1000
1001 999  1001
7 3
0 0 0
1 0 0
1 2 0
1 2 3
-1000 1000 -1000
-1001 1001 -1000
```

```
-1001 999  -1001
7 4
0 0 0
1 0 0
1 2 0
1 2 3
1000 -1000 1000
1001 -1001 1000
1001 -999  1001
0 0
```

the output should be

```
3
6
9
```

Problem G: You Win!

You just achieved the High Score on your favorite video game! Now, you get to enter your name! You have to use the controller to enter your name, which can be awkward.

Here's how it works:

- There are only the 26 capital letters, in order. There are no numbers, spaces, lower case letters, or any other characters.
- Pushing UP or DOWN moves the currently selected letter one letter forward or backward in the alphabet. It starts at A, and moving down from A wraps around to Z. Moving up from Z brings you back to A.
- Pushing LEFT or RIGHT moves the cursor one letter in the current name. Note that once the cursor is at either end of the current name, it cannot move any further in that direction.

  The currently selected letter is retained as you move around in the name.
- Pushing the FIRE button adds the current letter to the name.

For example, consider the name 'ALMA'. One way you could type 'ALMA' is like this (| denotes the position of the cursor):

Action	# of Pushes	Name	Current Letter
FIRE	1	A|	A
UP	11	A|	L
FIRE	1	AL|	L
UP	1	AL|	M
FIRE	1	ALM|	M
DOWN	12	ALM|	A
FIRE	1	ALMA|	A

This would take 28 button pushes. However, consider typing 'ALMA' like this:

Action	# of Pushes	Name	Current Letter
FIRE	1	A|	A
FIRE	1	AA|	A
LEFT	1	A|A	A
UP	11	A|A	L
FIRE	1	AL|A	L
UP	1	AL|A	M
FIRE	1	ALM|A	M

This takes only 17 button pushes.

Given a name, what is the fewest number of button pushes needed to enter that name? Assume that the current selected letter starts with A, and that it doesn't matter where the cursor ends up when you are done.

Input

There will be multiple test cases in the input. Each test case will consist of a single string on its own line, with from 1 to 20 capital letters, representing a name that must be entered into the High Score list.

End of input will be indicated by a line with a single 0.

Output

For each test case, output a single integer representing the smallest number of operations needed to enter the name.

Example

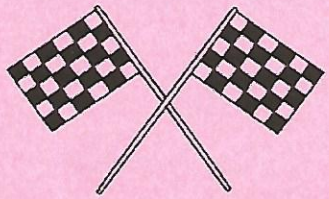
Given the input

```
ALMA
YES
0
```

the output should be

```
17
21
```

Problem H: Electric Car Rally



In an attempt to demonstrate the practicality of electric cars, ElecCarCo is sponsoring a cross-country road rally. There are n stations for the rally where cars may check in and charge their batteries.

The rally may require multiple days of travel. Each car can travel 4 hours (240 minutes) between charges. A car must be plugged into a charger for 2 minutes for each minute of travel time. Cars start the rally at noon on the first day, fully charged. Cars are permitted remain at a station even after they are fully charged, and there are enough chargers at each station to serve all cars.

It is only possible to drive directly between select pairs of stations. Furthermore, variations in traffic condition, road conditions, availability of HOV lanes, etc., result in different travel times along each route depending upon the time of day at which travel along that route begins.

The winner is the first car to reach station $n-1$, starting from station 0. Other than that starting and ending stations, cars may pass through the stations in any order, and need not visit all stations to complete the course.

Write a program to determine the earliest time, expressed as the total number of minutes elapsed since the start of the rally, at which a car could reach the final station.

Input

There will be multiple test cases in the input. Each test starts with a line containing n, $1 \leq n \leq 500$, the number of stations, and m, $1 \leq m \leq 1000$, the number of connecting road segments.

This is followed by m blocks, each describing one road segment. A road segment block has the following structure:

- Each block begins with a single line containing two integers, a and b ($0 \leq a, b \leq n-1$, $a \neq b$). These numbers are the two checkpoints connected by that segment. The connections are undirected: a segment permitting travel from checkpoint a to checkpoint b will also allow travel from checkpoint b to checkpoint a.

- This is followed by $1 \ldots 20$ "travel lines" describing travel times. Each of the travel lines contains 3 integers, *Start*, *Stop*, and *Time*. *Start* and *Stop* ($0 \leq Start < Stop \leq 1439$) are the time of day (expressed in minutes since midnight) described by this line, and *Time* ($0 < Time < 1000$) is the travel time, in minutes, required to traverse this road segment if travel begins at any time in the range [*Start*...*Stop*], inclusive.

  The first travel line in a block will have a start time of 0 (midnight, or 00:00). The final travel line in a block will have a stop time of 1439 (i.e., 23:59, or 1 minute less than 24 hours times 60 minutes). Adjacent travel lines in the input will be arranged in order, and the start time of any line after the first is one higher than the stop time of the preceding line. The travel lines will cover all times from 00:00 to 23:59.

- End of input will be indicated by a line with two 0s.

- All test cases will describe a course that can be completed by the cars.

Output

For each test case, output a single integer representing the smallest number of minutes needed to complete the rally.

Example

Given the input

```
4 4
0 1
0 1439 100
0 2
0 1439 75
1 3
0 720 150
721 824 100
825 1000 75
1001 1439 150
2 3
0 1439 150
3 2
0 1
0 10 200
11 1439 300
1 2
0 10 200
11 1439 300
0 0
```

the output should be

```
180
2360
```