Problem A
A Mazing!
Time limit: 5 seconds

A maze consists of a collection of equal sized square cells, where any or all of the sides may be a wall or a door. The maze may have no exit or multiple exits. Cells are typically arranged so that they share sides with other cells as shown in the four sample mazes in Figure A.1. From each cell you may attempt to move in one of four directions.

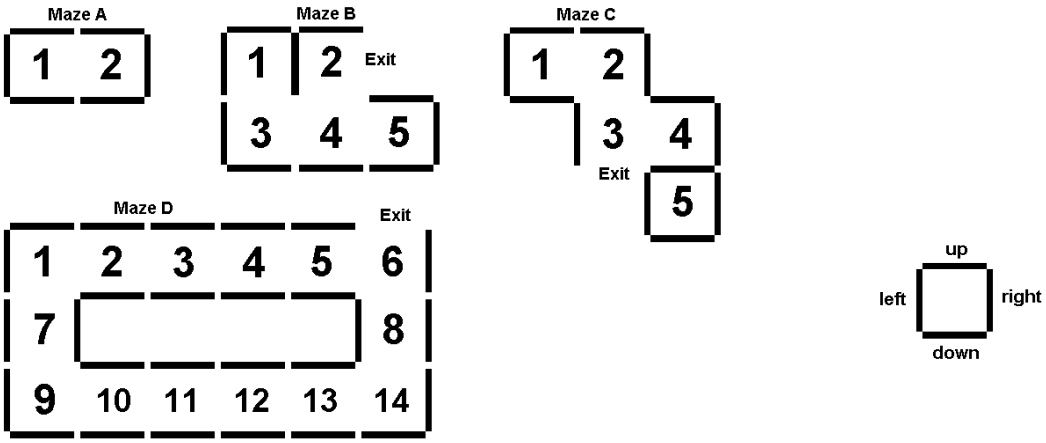

Figure A.1: Some maze examples and the four directions of movement.

For each maze, the starting point is someplace in the maze (for example, any of the numbered cells in the samples above). In the samples above:

- Maze A has no way out.
- Maze B has an exit (solution) to the right of cell 2.
- Maze C has an exit down from cell 3, unless the starting point is cell 5, in which case there is no way out.
- Maze D has an exit up from cell 6.

For example, using Maze D above, if the starting point is cell 9, one possible set of directions to get to the exit would be: `right`, `right`, `right`, `right`, `right`, `up`, `up`, `up`.

For this problem, you will write a program that finds an exit to a maze. Your program must operate *interactively*. That is, your program will make a move by providing a direction (`right`, `down`, `left` or `up`), and the judging software will send back one of three possible responses:

1. **`wall`** – indicates that a wall is there and you cannot proceed in that direction.
2. **`ok`** – indicates that there is door there and you proceeded in that direction to the neighboring cell.
3. **`solved`** – indicates that you have successfully found an exit to the maze.

If your program determines there is no way out of the maze, you should send the precise string "`no way out`" (without the quotes) instead of a direction. If there is in fact no way out of the maze, you will receive a **solved** reply.

After receiving a **solved** reply, your program should exit immediately.

Your program will receive a Wrong Answer judgement if any of the following occur:

1. Your program sends "`no way out`", even though there is a way out.
2. Your program makes the same move (direction) from the same cell twice.

Interaction

The input your program receives is a function of the output it generates. The first thing your program must do when it starts up is to send its first move (`up`, `down`, `right` or `left`). Each move your program makes is on its own line. It should then wait for a line of response on standard input. The response is one of **wall**, **ok**, or **solved**. Your program should then make another move based on the response it received as discussed above. This process repeats until your program receives a **solved** response or produces a wrong output.

It is guaranteed that the maze will not be larger than 100 rows by 100 columns.

Note: Do not forget to flush output buffers after each write. Consult the Addendum to Judging Notes for more details.

Read	Sample Interaction 1	Write
		down
wall		
		right
wall		
		left
wall		
		up
ok		
		right
ok		
		down
ok		
		down
wall		
		right
wall		
		left
wall		
		up
ok		
		right
solved		

Problem B
Catering
Time limit: 5 seconds

Picture from Wikimedia Commons

Paul owns a catering company and business is booming. The company has k catering teams, each in charge of one set of catering equipment. Every week, the company accepts n catering requests for various events. For every request, they send a catering team with their equipment to the event location. The team delivers the food, sets up the equipment, and instructs the host on how to use the equipment and serve the food. After the event, the host is responsible for returning the equipment back to Paul's company.

Unfortunately, in some weeks the number of catering teams is less than the number of requests, so some teams may have to be used for more than one event. In these cases, the company cannot wait for the host to return the equipment and must keep the team on-site to move the equipment to another location. The company has an accurate estimate of the cost to move a set of equipment from any location to any other location. Given these costs, Paul wants to prepare an Advance Catering Map to service the requests while minimizing the total moving cost of equipment (including the cost of the first move), even if that means not using all the available teams. Paul needs your help to write a program to accomplish this task. The requests are sorted in ascending order of their event times and they are chosen in such a way that for any $i < j$, there is enough time to transport the equipment used in the i^{th} request to the location of the j^{th} request.

Input

The first line of input contains two integers n ($1 \leq n \leq 100$) and k ($1 \leq k \leq 100$) which are the number of requests and the number of catering teams, respectively. Following that are n lines, where the i^{th} line contains $n - i + 1$ integers between 0 and 1 000 000 inclusive. The j^{th} number in the i^{th} line is the cost of moving a set of equipment from location i to location $i + j$. The company is at location 1 and the n requests are at locations 2 to $n + 1$.

Output

Display the minimum moving cost to service all requests. (This amount does not include the cost of moving the equipment back to the catering company.)

Sample Input 1	Sample Output 1
3 2 40 30 40 50 10 50	80

Sample Input 2	Sample Output 2
3 2 10 10 10 20 21 21	40

Problem C
Comma Sprinkler
Time limit: 4 seconds

As practice will tell you, the English rules for comma placement are complex, frustrating, and often ambiguous. Many people, even the English, will, in practice, ignore them, and, apply custom rules, or, no rules, at all.

Photo by Tanya Hart. Yarn pattern by Morgen Dämmerung.

Doctor Comma Sprinkler solved this issue by developing a set of rules that sprinkles commas in a sentence with no ambiguity and little simplicity. In this problem you will help Dr. Sprinkler by producing an algorithm to automatically apply her rules.

Dr. Sprinkler's rules for adding commas to an existing piece of text are as follows:

1. If a word anywhere in the text is preceded by a comma, find all occurrences of that word in the text, and put a comma before each of those occurrences, except in the case where such an occurrence is the first word of a sentence or already preceded by a comma.
2. If a word anywhere in the text is succeeded by a comma, find all occurrences of that word in the text, and put a comma after each of those occurrences, except in the case where such an occurrence is the last word of a sentence or already succeeded by a comma.
3. Apply rules 1 and 2 repeatedly until no new commas can be added using either of them.

As an example, consider the text

```
please sit spot. sit spot, sit. spot here now here.
```

Because there is a comma after `spot` in the second sentence, a comma should be added after `spot` in the third sentence as well (but not the first sentence, since it is the last word of that sentence). Also, because there is a comma before the word `sit` in the second sentence, one should be added before that word in the first sentence (but no comma is added before the word `sit` beginning the second sentence because it is the first word of that sentence). Finally, notice that once a comma is added after `spot` in the third sentence, there exists a comma before the first occurrence of the word `here`. Therefore, a comma is also added before the other occurrence of the word `here`. There are no more commas to be added so the final result is

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Input

The input contains one line of text, containing at least 2 characters and at most 1 000 000 characters. Each character is either a lowercase letter, a comma, a period, or a space. We define a *word* to be a maximal sequence of letters within the text. The text adheres to the following constraints:

- The text begins with a word.
- Between every two words in the text, there is either a single space, a comma followed by a space, or a period followed by a space (denoting the end of a sentence and the beginning of a new one).
- The last word of the text is followed by a period with no trailing space.

Output

Display the result after applying Dr. Sprinkler's algorithm to the original text.

Sample Input 1

```
please sit spot. sit spot, sit. spot here now here.
```

Sample Output 1

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Sample Input 2

```
one, two. one tree. four tree. four four. five four. six five.
```

Sample Output 2

```
one, two. one, tree. four, tree. four, four. five, four. six five.
```

Problem D
Go with the Flow
Time limit: 7 seconds

In typesetting, a "river" is a string of spaces formed by gaps between words that extends down several lines of text. For instance, Figure D.1 shows several examples of rivers highlighted in red (text is intentionally blurred to make the rivers more visible).

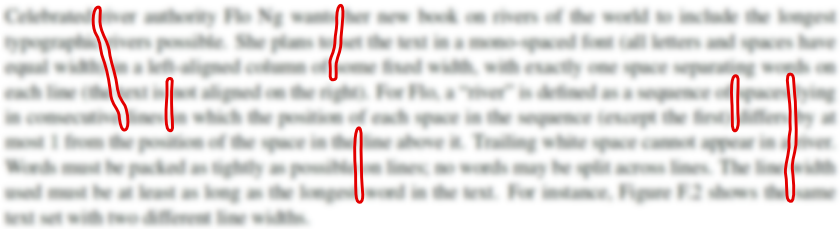

Figure D.1: Examples of rivers in typeset text.

Celebrated river authority Flo Ng wants her new book on rivers of the world to include the longest typographic rivers possible. She plans to set the text in a mono-spaced font (all letters and spaces have equal width) in a left-aligned column of some fixed width, with exactly one space separating words on each line (the text is not aligned on the right). For Flo, a "river" is defined as a sequence of spaces lying in consecutive lines in which the position of each space in the sequence (except the first) differs by at most 1 from the position of the space in the line above it. Trailing white space cannot appear in a river. Words must be packed as tightly as possible on lines; no words may be split across lines. The line width used must be at least as long as the longest word in the text. For instance, Figure D.2 shows the same text set with two different line widths.

Line width 14: River of length 4	Line width 15: River of length 5
The Yangtze is|	The Yangtze is |
the third |	the third |
longest river |	longest*river |
in*Asia and |	in Asia*and the|
the*longest in|	longest*in the |
the*world to |	world to*flow |
flow*entirely |	entirely*in one|
in one country|	country |

Figure D.2: Longest rivers (*) for two different line widths.

Given a text, you have been tasked with determining the line width that produces the longest river of spaces for that text.

Input

The first line of input contains an integer n ($2 \leq n \leq 2\,500$) specifying the number of words in the text. The following lines of input contain the words of text. Each word consists only of lowercase and uppercase letters, and words on the same line are separated by a single space. No word exceeds 80 characters.

Output

Display the line width for which the input text contains the longest possible river, followed by the length of the longest river. If more than one line width yields this maximum, display the shortest such line width.

Sample Input 1	Sample Output 1
21 The Yangtze is the third longest river in Asia and the longest in the world to flow entirely in one country	15 5

Sample Input 2	Sample Output 2
25 When two or more rivers meet at a confluence other than the sea the resulting merged river takes the name of one of those rivers	21 6

Problem E
Guess the Number
Time limit: 1 second

I am thinking of a number between 1 and 1 000; can you guess what it is? Given a guess, I will tell you whether it is too low, too high, or correct. But you only get 10 guesses per game, so use them wisely!

Interaction

We will play several games. The first thing your submission should do is read an integer n (where $1 \leq n \leq 100$), indicating the number of games that follow.

For each game, your submission should output guesses for the correct number, in the form of an integer between 1 and 1 000 on a line on its own.

After each guess, your submission should read a response on standard input. This response is a line with one of the following:

- "`lower`" if the number I am thinking of is lower than your guess;
- "`higher`" if the number I am thinking of is higher than your guess; or
- "`correct`" if your guess is correct.

After guessing correctly, you should move on to the next game (if any remain), or exit immediately (otherwise). Within each game, if you guess incorrectly 10 times, you won't get any more chances.

Note: Do not forget to flush output buffers after each write. Consult the Addendum to Judging Notes for more details.

Read	Sample Interaction 1	Write
2		
	800	
lower		
	300	
lower		
	5	
lower		
	2	
higher		
	4	
correct		
	999	
higher		
	1000	
correct		

This page is intentionally left blank.

Problem F
Pollution Solution
Time limit: 1 second

As an employee of Aqueous Contaminate Management, you must monitor the pollution that gets dumped (sometimes accidentally, sometimes purposefully) into rivers, lakes and oceans. One of your jobs is to measure the impact of the pollution on various ecosystems in the water such as coral reefs, spawning grounds, and so on.

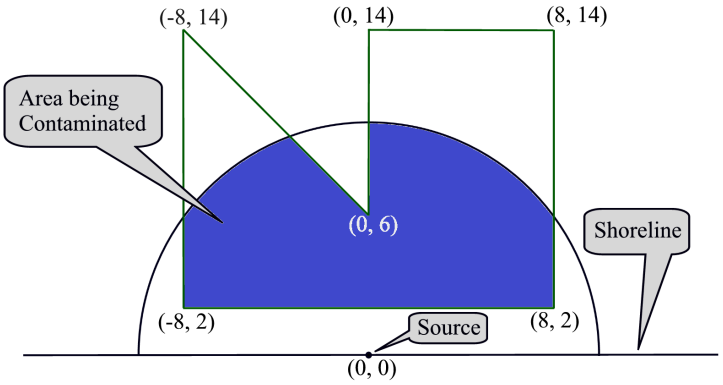

Figure F.1: Illustration of Sample Input 1.

The model you use in your analysis is illustrated in Figure F.1. The shoreline (the horizontal line in the figure) lies on the x-axis with the source of the pollution located at the origin (0,0). The spread of the pollution into the water is represented by the semicircle, and the polygon represents the ecosystem of concern. You must determine the area of the ecosystem that is contaminated, represented by the dark blue region in the figure.

Input

The input consists of a single test case. A test case starts with a line containing two integers n and r, where $3 \leq n \leq 100$ is the number of vertices in the polygon and $1 \leq r \leq 1\,000$ is the radius of the pollution field. This is followed by n lines, each containing two integers x_i, y_i, giving the coordinates of the polygon vertices in counter-clockwise order, where $-1\,500 \leq x_i \leq 1\,500$ and $0 \leq y_i \leq 1\,500$. The polygon does not self-intersect or touch itself. No vertex lies on the circle boundary.

Output

Display the area of the polygon that falls within the semicircle centered at the origin with radius r. Give the result with an absolute error of at most 10^{-3}.

Sample Input 1	Sample Output 1
6 10 -8 2 8 2 8 14 0 14 0 6 -8 14	101.576437872

Problem G
A Mazing! 2
Time limit: 5 seconds

A maze consists of a collection of equal sized square cells, where any or all of the sides may be a wall or a door. The maze may have no exit or multiple exits. Cells are typically arranged so that they share sides with other cells as shown in the four sample mazes in Figure G.1. From each cell you may attempt to move in one of four directions.

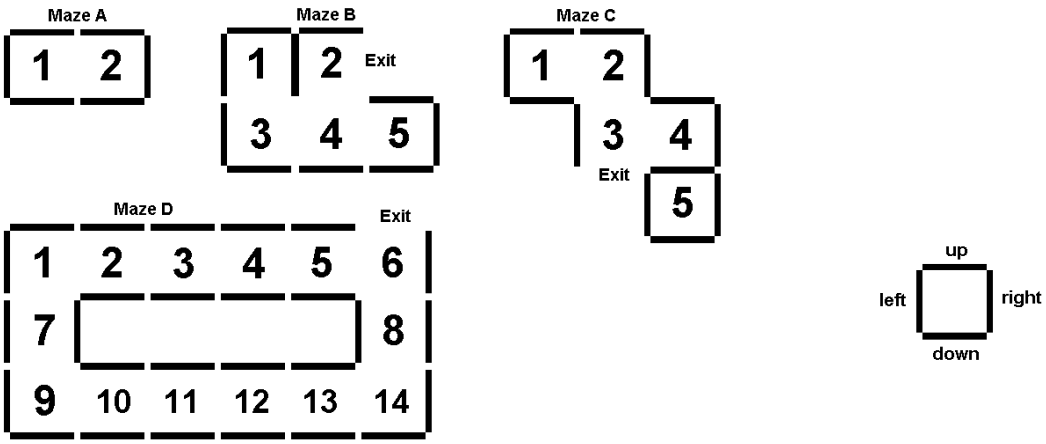

Figure G.1: Some maze examples and the four directions of movement.

For each maze, the starting point is someplace in the maze (for example, any of the numbered cells in the samples above). In the samples above:

- Maze A has no way out.
- Maze B has an exit (solution) to the right of cell 2.
- Maze C has an exit down from cell 3, unless the starting point is cell 5, in which case there is no way out.
- Maze D has an exit up from cell 6.

For example, using Maze D above, if the starting point is cell 9, one possible set of directions to get to the exit would be: `right`, `right`, `right`, `right`, `right`, `up`, `up`, `up`.

For this problem, you will write a program that finds an exit to a maze. Your program must operate *interactively*. That is, your program will make a move by providing a direction (`right`, `down`, `left` or `up`), and the judging software will send back one of three possible responses:

1. **`wall`** – indicates that a wall is there and you cannot proceed in that direction.
2. **`ok`** – indicates that there is door there and you proceeded in that direction to the neighboring cell.
3. **`solved`** – indicates that you have successfully found an exit to the maze.

If your program determines there is no way out of the maze, you should send the precise string "`no way out`" (without the quotes) instead of a direction. If there is in fact no way out of the maze, you will receive a **solved** reply.

After receiving a **solved** reply, your program should exit immediately.

Your program will receive a Wrong Answer judgement if any of the following occur:

1. Your program sends "`no way out`", even though there is a way out.
2. Your program makes the same move (direction) from the same cell twice.

Interaction

The input your program receives is a function of the output it generates. The first thing your program must do when it starts up is to send its first move (`up`, `down`, `right` or `left`). Each move your program makes is on its own line. It should then wait for a line of response on standard input. The response is one of **wall**, **ok**, or **solved**. Your program should then make another move based on the response it received as discussed above. This process repeats until your program receives a **solved** response or produces a wrong output.

It is guaranteed that the maze will not be larger than 100 rows by 100 columns.

Note: Do not forget to flush output buffers after each write. Consult the Addendum to Judging Notes for more details.

Read	Sample Interaction 1	Write
		down
wall		
		right
wall		
		left
wall		
		up
ok		
		right
ok		
		down
ok		
		down
wall		
		right
wall		
		left
wall		
		up
ok		
		right
solved		

Problem H
Catering 2
Time limit: 5 seconds

Picture from Wikimedia Commons

Paul owns a catering company and business is booming. The company has k catering teams, each in charge of one set of catering equipment. Every week, the company accepts n catering requests for various events. For every request, they send a catering team with their equipment to the event location. The team delivers the food, sets up the equipment, and instructs the host on how to use the equipment and serve the food. After the event, the host is responsible for returning the equipment back to Paul's company.

Unfortunately, in some weeks the number of catering teams is less than the number of requests, so some teams may have to be used for more than one event. In these cases, the company cannot wait for the host to return the equipment and must keep the team on-site to move the equipment to another location. The company has an accurate estimate of the cost to move a set of equipment from any location to any other location. Given these costs, Paul wants to prepare an Advance Catering Map to service the requests while minimizing the total moving cost of equipment (including the cost of the first move), even if that means not using all the available teams. Paul needs your help to write a program to accomplish this task. The requests are sorted in ascending order of their event times and they are chosen in such a way that for any $i < j$, there is enough time to transport the equipment used in the i^{th} request to the location of the j^{th} request.

Input

The first line of input contains two integers n ($1 \leq n \leq 100$) and k ($1 \leq k \leq 100$) which are the number of requests and the number of catering teams, respectively. Following that are n lines, where the i^{th} line contains $n - i + 1$ integers between 0 and 1 000 000 inclusive. The j^{th} number in the i^{th} line is the cost of moving a set of equipment from location i to location $i + j$. The company is at location 1 and the n requests are at locations 2 to $n + 1$.

Output

Display the minimum moving cost to service all requests. (This amount does not include the cost of moving the equipment back to the catering company.)

Sample Input 1	Sample Output 1
3 2 40 30 40 50 10 50	80

Sample Input 2	Sample Output 2
3 2 10 10 10 20 21 21	40

Problem I
Comma Sprinkler 2
Time limit: 4 seconds

As practice will tell you, the English rules for comma placement are complex, frustrating, and often ambiguous. Many people, even the English, will, in practice, ignore them, and, apply custom rules, or, no rules, at all.

Photo by Tanya Hart. Yarn pattern by Morgen Dämmerung.

Doctor Comma Sprinkler solved this issue by developing a set of rules that sprinkles commas in a sentence with no ambiguity and little simplicity. In this problem you will help Dr. Sprinkler by producing an algorithm to automatically apply her rules.

Dr. Sprinkler's rules for adding commas to an existing piece of text are as follows:

1. If a word anywhere in the text is preceded by a comma, find all occurrences of that word in the text, and put a comma before each of those occurrences, except in the case where such an occurrence is the first word of a sentence or already preceded by a comma.
2. If a word anywhere in the text is succeeded by a comma, find all occurrences of that word in the text, and put a comma after each of those occurrences, except in the case where such an occurrence is the last word of a sentence or already succeeded by a comma.
3. Apply rules 1 and 2 repeatedly until no new commas can be added using either of them.

As an example, consider the text

```
please sit spot. sit spot, sit. spot here now here.
```

Because there is a comma after `spot` in the second sentence, a comma should be added after `spot` in the third sentence as well (but not the first sentence, since it is the last word of that sentence). Also, because there is a comma before the word `sit` in the second sentence, one should be added before that word in the first sentence (but no comma is added before the word `sit` beginning the second sentence because it is the first word of that sentence). Finally, notice that once a comma is added after `spot` in the third sentence, there exists a comma before the first occurrence of the word `here`. Therefore, a comma is also added before the other occurrence of the word `here`. There are no more commas to be added so the final result is

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Input

The input contains one line of text, containing at least 2 characters and at most 1 000 000 characters. Each character is either a lowercase letter, a comma, a period, or a space. We define a *word* to be a maximal sequence of letters within the text. The text adheres to the following constraints:

- The text begins with a word.
- Between every two words in the text, there is either a single space, a comma followed by a space, or a period followed by a space (denoting the end of a sentence and the beginning of a new one).
- The last word of the text is followed by a period with no trailing space.

Output

Display the result after applying Dr. Sprinkler's algorithm to the original text.

Sample Input 1

```
please sit spot. sit spot, sit. spot here now here.
```

Sample Output 1

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Sample Input 2

```
one, two. one tree. four tree. four four. five four. six five.
```

Sample Output 2

```
one, two. one, tree. four, tree. four, four. five, four. six five.
```

Problem J
Go with the Flow 2
Time limit: 7 seconds

In typesetting, a "river" is a string of spaces formed by gaps between words that extends down several lines of text. For instance, Figure J.1 shows several examples of rivers highlighted in red (text is intentionally blurred to make the rivers more visible).

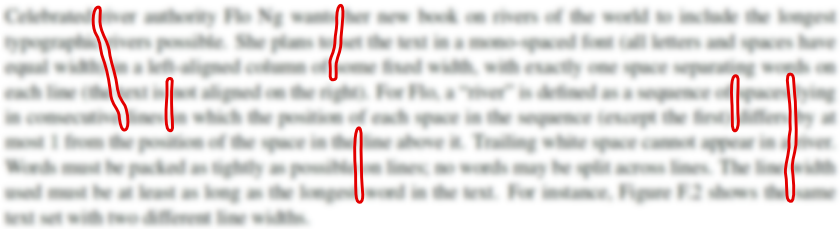

Figure J.1: Examples of rivers in typeset text.

Celebrated river authority Flo Ng wants her new book on rivers of the world to include the longest typographic rivers possible. She plans to set the text in a mono-spaced font (all letters and spaces have equal width) in a left-aligned column of some fixed width, with exactly one space separating words on each line (the text is not aligned on the right). For Flo, a "river" is defined as a sequence of spaces lying in consecutive lines in which the position of each space in the sequence (except the first) differs by at most 1 from the position of the space in the line above it. Trailing white space cannot appear in a river. Words must be packed as tightly as possible on lines; no words may be split across lines. The line width used must be at least as long as the longest word in the text. For instance, Figure J.2 shows the same text set with two different line widths.

Line width 14: River of length 4	Line width 15: River of length 5
The Yangtze is|	The Yangtze is |
the third |	the third |
longest river |	longest*river |
in*Asia and |	in Asia*and the|
the*longest in|	longest*in the |
the*world to |	world to*flow |
flow*entirely |	entirely*in one|
in one country|	country |

Figure J.2: Longest rivers (*) for two different line widths.

Given a text, you have been tasked with determining the line width that produces the longest river of spaces for that text.

Input

The first line of input contains an integer n ($2 \leq n \leq 2\,500$) specifying the number of words in the text. The following lines of input contain the words of text. Each word consists only of lowercase and uppercase letters, and words on the same line are separated by a single space. No word exceeds 80 characters.

Output

Display the line width for which the input text contains the longest possible river, followed by the length of the longest river. If more than one line width yields this maximum, display the shortest such line width.

Sample Input 1	Sample Output 1
21 The Yangtze is the third longest river in Asia and the longest in the world to flow entirely in one country	15 5

Sample Input 2	Sample Output 2
25 When two or more rivers meet at a confluence other than the sea the resulting merged river takes the name of one of those rivers	21 6

Problem K
Guess the Number 2
Time limit: 1 second

I am thinking of a number between 1 and 1 000; can you guess what it is? Given a guess, I will tell you whether it is too low, too high, or correct. But you only get 10 guesses per game, so use them wisely!

Interaction

We will play several games. The first thing your submission should do is read an integer n (where $1 \leq n \leq 100$), indicating the number of games that follow.

For each game, your submission should output guesses for the correct number, in the form of an integer between 1 and 1 000 on a line on its own.

After each guess, your submission should read a response on standard input. This response is a line with one of the following:

- "`lower`" if the number I am thinking of is lower than your guess;
- "`higher`" if the number I am thinking of is higher than your guess; or
- "`correct`" if your guess is correct.

After guessing correctly, you should move on to the next game (if any remain), or exit immediately (otherwise). Within each game, if you guess incorrectly 10 times, you won't get any more chances.

Note: Do not forget to flush output buffers after each write. Consult the Addendum to Judging Notes for more details.

Sample Interaction 1

Read	Write
2	
	800
lower	
	300
lower	
	5
lower	
	2
higher	
	4
correct	
	999
higher	
	1000
correct	

This page is intentionally left blank.

Problem L
Pollution Solution 2
Time limit: 1 second

As an employee of Aqueous Contaminate Management, you must monitor the pollution that gets dumped (sometimes accidentally, sometimes purposefully) into rivers, lakes and oceans. One of your jobs is to measure the impact of the pollution on various ecosystems in the water such as coral reefs, spawning grounds, and so on.

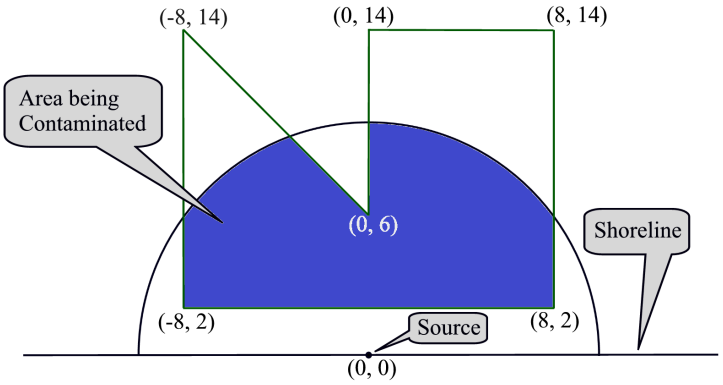

Figure L.1: Illustration of Sample Input 1.

The model you use in your analysis is illustrated in Figure L.1. The shoreline (the horizontal line in the figure) lies on the x-axis with the source of the pollution located at the origin (0,0). The spread of the pollution into the water is represented by the semicircle, and the polygon represents the ecosystem of concern. You must determine the area of the ecosystem that is contaminated, represented by the dark blue region in the figure.

Input

The input consists of a single test case. A test case starts with a line containing two integers n and r, where $3 \leq n \leq 100$ is the number of vertices in the polygon and $1 \leq r \leq 1\,000$ is the radius of the pollution field. This is followed by n lines, each containing two integers x_i, y_i, giving the coordinates of the polygon vertices in counter-clockwise order, where $-1\,500 \leq x_i \leq 1\,500$ and $0 \leq y_i \leq 1\,500$. The polygon does not self-intersect or touch itself. No vertex lies on the circle boundary.

Output

Display the area of the polygon that falls within the semicircle centered at the origin with radius r. Give the result with an absolute error of at most 10^{-3}.

Sample Input 1	Sample Output 1
6 10 -8 2 8 2 8 14 0 14 0 6 -8 14	101.576437872