Problem	IT	WINDOWS	AIRPLANE	ELBOW
Executable file	`it.exe`	`windows.exe`	`airplane.exe`	`elbow.exe`

Source file	`it.pas`
	`it.c`
	`it.cpp`
		`windows.pas`
		`windows.c`
		`windows.cpp`
			`airplane.pas`
			`airplane.c`
			`airplane.cpp`
				`elbow.pas`
				`elbow.c`
				`elbow.cpp`

Input file	`it.in`	`windows.in`	`airplane.in`	`elbow.in`
Output file	`it.out`	`windows.out`	`airplane.out`	`elbow.out`
Time limit per test	10 seconds	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10	10
Points per test	3	4	6	7
Total points	**30**	**40**	**60**	**70**
	200			

Mirko and Slavko are playing one well-known interesting game.

Table for that game doesn't have some specific name, so let's called it 'it'. 'It' stands in vertical position and it has 6 rows and 7 columns. The players are throwing their coins in 'it' one after another. When somebody put his coin in some column, coin is falling until the lowest free row. Each coin is taking exactly one position.

Each player has 21 coins and **Mirko** is playing **first**. Mirko has yellow coins, and Slavko has red coins.

The aim of the game is connect **four coins in the sequence** in horizontal, vertical or diagonal direction. We called that situation the winning situation.

When Mirko and Slavko are playing the game, they don't look if the game is (and when is) over, they just play until there are no coins to throw in front of them. Because of that, they decide to write down on the piece of paper all their moves and to let somebody smarter to find out who is the winner of the game.

Write the program that will decide **who** won and in **which** move. In the case that winning situation happened more than once in the game, **first of them** is making winner. However, the game can finish without the winner.

Input data

Input file consists of 21 lines.

In i-th line are two integers M_i and S_i, separated with space. Number M_i stands for number of column in which Mirko throw his coin in i-th move, and number S_i number of column in which Slavko throw his coin in i-th move.

Columns are denoted with numbers from 1 to N.

Output data

In first and only line of input file you must write '`mirko M`' or '`slavko M`' if Mirko or Slavko won in M-th move or word '`remi`' if there are no winner in the game.

Examples

it.in

```
2 2
5 2
3 7
6 1
4 6
3 1
3 5
3 3
6 3
2 5
4 1
6 2
2 5
7 5
1 7
4 4
4 1
7 6
1 7
7 5
6 4
```

it.out

```
mirko 5
```

it.in

```
1 1
2 4
6 4
5 1
1 2
7 4
3 4
7 7
4 6
3 5
2 4
6 1
7 3
6 3
6 1
6 3
2 5
7 3
2 7
2 5
5 5
```

it.out

```
slavko 7
```

it.in

```
1 2
1 2
1 2
2 1
2 1
2 1
3 4
3 4
3 4
4 3
4 3
4 3
5 6
5 6
5 6
6 5
6 5
6 5
7 7
7 7
7 7
```

it.out

```
remi
```

WINDOWS

In popular operating system we opened lots of windows.

Each window is rectangle consisting of small squares (of dimension 1 x 1).

Newly opened windows, depending of its position and size, **can** (partially or totally) cover some earlier opened windows.

We can close window with mouse if we click on its **upper right** small square, which in that moment **must be visible**. Square of some window is visible if there is no later opened window, which is not yet closed.

Write the program that will calculate the **minimal** number of mouse clicks to **close** the window that we opened **first**.

Input data

In first line of input file there is integer N, number of windows, $1 \leq N \leq 100$.

In each of the next N lines there are 4 integers R1, S1, R2 and S2, separated with spaces, $1 \leq R1 \leq R2 \leq 10000$, $1 \leq S1 \leq S2 \leq 10000$. Numbers R1 and S1 stand for row and column of the screen in which is **upper-left** square of that window, and R2 and S2 stand for row and column of the screen in which is **lower-right** square of that windows. Windows are opened in the **same** order like they appears in the input file.

We imagine screen as it consits of rows and columns of small squares. Rows are numerated from the top to the bottom, columns from the left to the right, and upper-left square on the screen is in the first row and first columnt.

Output data

In first and only line of output file you should write the minimal number of mouse clicks.

Examples

windows.in

```
3
3 1 6 4
1 2 4 6
2 3 5 5
```

windows.out

```
3
```

windows.in

```
3
4 1 6 3
2 2 5 5
1 4 3 6
```

windows.out

```
3
```

windows.in

```
3
3 3 4 4
1 1 2 2
5 5 6 6
```

windows.out

```
1
```

AIRPLANE

Imagine the airplane in which exist only one corridor through the rows of seats and passengers, which are entering in the corridor and looking for their seats.

Each passenger must seat on his seat, and entrance to airplane is on the begging, just in front of first row of seats. Passengers are entering one after another, without any unnecessary stops.

Passenger is walking through corridor in the direction of his seat and in each row he stay for **exactly one** second (or more, if in front of them is another passenger) until he arrive to his row and then he stay in his row for **five** seconds (because of putting luggage). In each second in the corridor at one row can be **only one** passenger.

Write the program which will calculate how much time is needed that corridor become **free** i.e. that all passengers seat on their seats.

Input data

In the first line of input file there is integer N, number of passengers, $1 \leq N \leq 1000$.

In (i+1)-th line is one integer R_i, number of row where the passenger number i must seat, $1 \leq R_i \leq 1000$.

Passengers are numerated with number from 1 to N, and they are entering in that order. Number of passengers who are seating in the same row is **not limited**.

Output data

In first and only line of output file you should write time (in seconds) that is needed that all the passengers seat on their seats.

Examples

airplane.in

```
1
3
```

airplane.out

```
7
```

airplane.in

```
2
3
3
```

airplane.out

```
12
```

airplane.in

```
4
4
4
1
5
```

airplane.out

```
19
```

ELBOW

Mirko and Slavko are playing very interesting game.

Each of them imagines one word, then they seat on the same computer and hitting each other with elbows typing their words on the keyboard.

After that, they look at the screen and they see that their words are so **mixed** that they cannot find which letter belongs to one and which to another of them.

Write the program that will decide for each letter from the screen if it belongs to Mirko's or Slavko's word.

Note: solution does not to be unique.

Input data

In first and second line of input file there are Mirko's and Slavko's word. Each word consists only from lowercase letters from English alphabet (a-z), and number of letters in each word will be less or equal than 150.

In third row there is word from the screen.

There will **always** be solution from the input data.

Output data

In first and only line of output file you should for each letter from the third word write down number '**1**' or number '**2**', so if we read letters from the third word on the positions denoted by number '**1**' we get **first** word, and if we read letters from the third word on the positions denoted by number '**2**' we get **second** word.

Examples

elbow.in	**elbow.in**	**elbow.in**
novine vesna novesvinena	tata mama mtatamaa	hsin sinh hsinhsin
elbow.out	**elbow.out**	**elbow.out**
11222111122	21112212	12222111