Problem 1

SMILE! :-)

September 19, 1982 — On this day, a Carnegie Mellon University computer science professor named Scott E. Fahlman posted a smiley face **:-)** on an electronic message board.

Not just content to remain a frozen smirk, it has learned to wink **;-)** and stick out its tongue **:-p**. Sometimes it gets sad **:-(** or confused **:-S**, and sometimes it's scared **=:-0** but most of the time it's cheerful **:-D**.

As it grew older, it had its first kiss **:-*** and smoked its first cigarette **:-Q** and even grew its hair out **&:-)**. It's shown its fashion sense, too, sporting sunglasses **B-)** and baseball caps **d:-)**, and even the occasional bow tie **:-)8**.

Your job is to count the number of original smiley faces in a single line of input. That is, the number of times the substring "**:-)**" occurs.

Input

Input consists of a single string of length *n*, ($2 \leq n \leq 80$). Assume the last character in the file will always be an end of line character.

Output

Output the total times the substring "`:-)`" occurs on a given line of input. The output should be formatted exactly as shown below.

Sample Input

```
From: <Fahlman@Cmu>:-D :-) Read it sideways::-)-) ;-)Happy:-)8 Coding!
```

Output Corresponding to Sample Input

```
Smiley Count = 3
```

Problem 2

Look and Say

The look and say sequence is defined as follows. Start with any string of decimal digits as the first element in the sequence. Each subsequent element is defined from the previous one by "verbally" describing the previous element. For example, the string 122344111 can be described as "one 1, two 2's, one 3, two 4's, three 1's". Therefore, the element that comes after 122344111 in the sequence is 1122132431. Similarly, the string 101 comes after 1111111111.

Notice that it is generally not possible to uniquely identify the previous element of a particular element. For example, a string of 112213243 1's also yields 1122132431 as the next element.

Input & Output

The first line of input will be a number *n*, ($2 \leq n \leq 20$), telling how many cases are to follow. The next *n*-lines consist of several cases. Each case consists of a line of up to 1000 digits. For each test case, print the string that follows the given string. Assume all input will consist of decimal digits 0 – 9.

Sample Input

```
3
122344111
1111111111
12345
```

Output Corresponding to Sample Input

```
1122132431
101
1112131415
```

Problem 3

Magic Clocks

Andy is simply fascinated by clocks! In fact, he finds it hard to focus on his grading because he stares at his Mickey Mouse clock all day! So, to inspire him to grade his computer science programs he has devised a method to keep him on task. He will only look at the clock at "magical" times. A magical time is a time at which the angle between the hour hand and the minute hand is equal to the angle between the minute hand and the second hand on Andy's Disney clock.

Andy is so entranced by his clock that he just can't wait for the next magical time. Your task is to calculate the soonest magical time, given some time in the day. An important feature of Andy's analog clock is that it is *not continuous*, i.e. it does not smoothly flow into the next time. When the secondhand ticks to 0, the minute hand *ticks* to the next minute, rather than continuously move. This is also the case with hours.

Input
The first line of input will be a number *n*, ($2 \leq n \leq 20$), telling how many lines of input are to follow. The next *n*-lines will contain a time in the `HH:MM:SS` format at the start of the line.

Output
For each time input, you should output "`The next magical time is` *s* `seconds away.`" If the time is already magical, then you should print, "`Look Andy, it's a magical time!`" If *s* is exactly one, print `second` rather than `seconds`. You may assume the next magical time is guaranteed to be within the next minute.

Sample Input
```
5
12:00:00
12:00:01
06:14:39
07:29:34
01:59:00
```

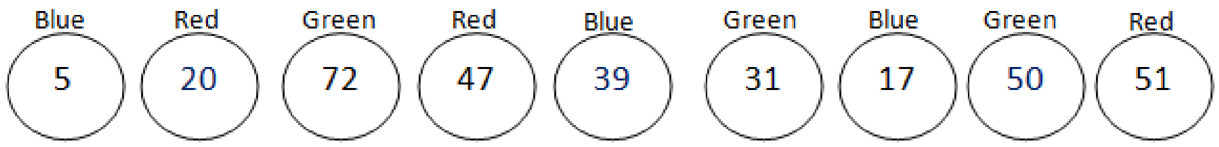
Output Corresponding to Sample Input
```
Look Andy, it's a magical time!
The next magical time is 59 seconds away.
The next magical time is 19 seconds away.
The next magical time is 1 second away.
The next magical time is 5 seconds away.
```

Problem 4

CCSC Lottery

This year CCSC has decided to find the winning numbers of the CCSC Lottery using a computer program. The winning numbers will be drawn from a bowl of balls. Each ball in this bowl has a color and a number. To decide the winning numbers, first we will construct chains of balls. Each chain should contain a ball from each color. That means if we have only three colors, then each chain should contain exactly three balls (one from each color).

For each such chain, the *value* of the chain is defined as the maximum difference between the numbers on the balls in the chain. For example, if we have a chain with three balls: Red with number 10, Green with 15, and Blue with 17, then the value of the chain will be 7 which is the maximum difference between 10, 15, and 17. Now the winning numbers should be constructed using the chain with the minimum value. The winning numbers should be written as the numbers from the balls in the chain in ascending order. Consider the following example.



Some of the chains and their corresponding values for this example will be as shown below.

5 (Blue) – 20 (Red) – 31 (Green) : Value 26
17 (Blue) – 20 (Red) – 31 (Green) : Value 14
20 (Red) – 31 (Green) – 39 (Blue) : Value 19
31 (Green) – 39 (Blue) – 47 (Red) : Value 16
39 (Blue) – 47 (Red) – 50 (Green) : Value 11

The last chain has the minimum value. So the winning number will be 39 47 50. You can assume that there will be exactly one chain with the minimum value.

Input
The first line of input will be *n* and *m* where *n* is the number of colors and *m* is the number of balls in each color. The next $n * m$ lines will describe each ball in the input in the following format.

Number   *Color*

Number and *Color* will contain 1 more spaces between them. To simplify the input, each color will be represented by a number starting with 0.

Output
Your program should output the winning numbers of the one winner. Write the numbers from the chain with minimum value in ascending order. Separate each number by a single blank space.

Sample Input

```
3    3
5    0
20   1
72   2
47   1
39   0
31   2
17   0
50   2
51   1
```

Output Corresponding to Sample Input

```
39 47 50
```