The 3n + 1 problem

Background

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

The Problem

Consider the following algorithm:

```
1.    input n
2.    print n
3.    if n = 1 then STOP
4.        if n is odd then $n \longleftarrow 3n+1$
5.        else $n \longleftarrow n/2$
6.    GOTO 2
```

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that 0 < n < 1,000,000 (and, in fact, for many more numbers than this.)

Given an input n, it is possible to determine the number of numbers printed (including the 1). For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16.

For any two numbers i and j you are to determine the maximum cycle length over all numbers between i and j.

The Input

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j.

You can assume that no operation overflows a 32-bit integer.

The Output

For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

Sample Input

```
1 10
100 200
201 210
900 1000
```

Sample Output

```
1 10 20
100 200 125
201 210 89
900 1000 174
```

Problem B: Minesweeper

The Problem

Have you ever played Minesweeper? It's a cute little game which comes within a certain Operating System which name we can't really remember. Well, the goal of the game is to find where are all the mines within a MxN field. To help you, the game shows a number in a square which tells you how many mines there are adjacent to that square. For instance, supose the following 4x4 field with 2 mines (which are represented by an * character):

```
*...
....
.*..
....
```

If we would represent the same field placing the hint numbers described above, we would end up with:

```
*100
2210
1*10
1110
```

As you may have already noticed, each square may have at most 8 adjacent squares.

The Input

The input will consist of an arbitrary number of fields. The first line of each field contains two integers n and m ($0 < n,m <= 100$) which stands for the number of lines and columns of the field respectively. The next n lines contains exactly m characters and represent the field. Each safe square is represented by an "." character (without the quotes) and each mine square is represented by an "*" character (also without the quotes). The first field line where n = m = 0 represents the end of input and should not be processed.

The Output

For each field, you must print the following message in a line alone:

```
Field #x:
```

Where x stands for the number of the field (starting from 1). The next n lines should contain the field with the "." characters replaced by the number of adjacent mines to that square. There must be an empty line between field outputs.

Sample Input

```
4 4
*...
```

```
....
.*..
....
3 5
**...
.....
.*...
0 0
```

Sample Output

```
Field #1:
*100
2210
1*10
1110

Field #2:
**100
33200
1*100
```

Question B - Contest Scoreboard

Think the contest score boards are wrong? Here's your chance to come up with the right rankings.

Contestants are ranked first by the number of problems solved (the more the better), then by decreasing amounts of penalty time. If two or more contestants are tied in both problems solved and penalty time, they are displayed in order of increasing team numbers.

A problem is considered solved by a contestant if any of the submissions for that problem was judged correct. Penalty time is computed as the number of minutes it took for the first correct submission for a problem to be received plus 20 minutes for each incorrect submission received prior to the correct solution. Unsolved problems incur no time penalties.

Input:

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

Input consists of a snapshot of the judging queue, containing entries from some or all of contestants 1 through 100 solving problems 1 through 9. Each line of input will consist of three numbers and a letter in the format

contestant problem time L

where L can be `C`, `I`, `R`, `U` or `E`. These stand for Correct, Incorrect, clarification Request, Unjudged and Erroneous submission. The last three cases do not affect scoring.

Lines of input are in the order in which submissions were received.

Output:

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Output will consist of a scoreboard sorted as previously described. Each line of output will contain a contestant number, the number of problems solved by the contestant and the time penalty accumulated by the contestant. Since not all of contestants 1-100 are actually participating, display only the contestants that have made a submission.

Sample Input:

```
1

1 2 10 I
3 1 11 C
```

```
1 2 19 R
1 2 21 C
1 1 25 C
```

Sample Output:

```
1 2 66
3 1 11
```