ACM International Collegiate Programming Contest
2000 East Central Regional Practice Contest
Case Western Reserve University
University of Cincinnati
November 10, 2000

Sponsored by IBM

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final. No cheating will be tolerated.
8. There are **four** questions to be completed in **one hour and twenty minutes**.

Problem A: Ride My Bicycle

Alan Chester Mason, worried about the overuse of fossil fuels and his own expanding waist, has decided to purchase a bicycle. The model he has decided on has 3 chain rings on the crank (the gears attached to the pedal mechanism) and 7 gears on the back wheel. There is a chain connecting one front chain ring to one back gear. This selection can be changed (via the derailleurs) to any combination of front chain ring and back gear desired. The ease of pedalling is determined by the ratio of teeth in the chosen chain ring and the chosen back ring. The smaller the ratio, the easier it is to pedal. Alan has many options for the number of teeth in each chain ring and each back gear and wishes to see what the gearing (that is, ratios) would be for various combinations. You are to write a program to help him out.

Input

Input for this program will be the description of one configuration of chain rings and back gears. Input will be specified on two lines. The first line will have three positive integers, separated by single spaces, in strictly increasing order giving the number of teeth on each chain ring. The second line will contain 7 positive integers, separated by spaces, in strictly increasing order giving the number of teeth on each back gear. We will refer to the chain rings as chain rings 1, 2 and 3 from smallest to largest. We refer to the back gears in a similar manner.

Output

You should generate 21 lines of output for this program. Each line should give information about one particular combination of chain ring and back ring and should be of the form

```
x.xx f b
```

where `f` is the number of the chain ring, `b` is the number of the back ring and `x.xx` is the ratio of (teeth in `f`)/(teeth in `b`) given to 2 decimal places. These numbers are separated by a single space. These lines are to be printed from smallest ratio to largest. If two (or more) ratios are equal, print the one with the smallest chain ring first. You may assume that the number of teeth are such that the ratios will all be less than 10.

Sample Input

```
32 42 52
12 15 17 18 21 24 28
```

Sample Output

```
1.14 1 7
1.33 1 6
1.50 2 7
1.52 1 5
1.75 2 6
```

```
1.78 1 4
1.86 3 7
1.88 1 3
2.00 2 5
2.13 1 2
2.17 3 6
2.33 2 4
2.47 2 3
2.48 3 5
2.67 1 1
2.80 2 2
2.89 3 4
3.06 3 3
3.47 3 2
3.50 2 1
4.33 3 1
```

Problem B: Counting Sheep

After a long night of coding, Charles Pearson Peterson is having trouble sleeping. This is not only because he is still thinking about the problem he is working on but also due to drinking too much java during the wee hours. This happens frequently, so Charles has developed a routine to count sheep. Not the animal, but the word. Specifically, he thinks of a list of words, many of which are close in spelling to "sheep", and then counts how many actually are the word "sheep". Charles is always careful to be case-sensitive in his matching, so "Sheep" is not a match. You are to write a program that helps Charles count "sheep".

Input

Input will consist of multiple problem instances. The first line will consist of a single positive integer $n \leq 20$, which is the number of problem instances. The input for each problem instance will be on two lines. The first line will consist of a positive integer $m \leq 10$ and the second line will consist of m words, separated by a single space and each containing no more than 10 characters.

Output

For each problem instance, you are to produce one line of output in the format:

```
Case i: This list contains n sheep.
```

The value of `i` is the number of the problem instance (we assume we start numbering at 1) and `n` is the number of times the word "sheep" appears in the list of words for that problem instance. Output lines should be separated by a single blank line.

Sample Input

```
4
5
shep sheeps sheep ship Sheep
7
sheep sheep SHEEP sheep shepe shemp seep
10
sheep sheep sheep sheep sheep sheep sheep sheep sheep sheep
4
shape buffalo ram goat
```

Sample Output

```
Case 1: This list contains 1 sheep.

Case 2: This list contains 3 sheep.

Case 3: This list contains 10 sheep.

Case 4: This list contains 0 sheep.
```

Problem C: Sequence

The sequence 1, 1010, 2012, 10021 may not look like an arithmetic sequence, but it is one in base 3. Likewise, the sequence 11, 33, 55 is clearly an arithmetic sequence in base 10, but it is also one in base 6. For this problem, you will be given a sequence of numbers and you must write an Arithmetic Confirmation Machine to determine the smallest base under which the numbers form an arithmetic sequence.

Input

Input will consist of multiple problem instances. The first line will contain a single integer $2 \leq n \leq 5$ indicating the number of values in the sequence. The next line will contain the n numbers in strictly increasing order, separated by a single blank. A value of $n = 0$ will terminate the input. All numbers will be positive and made up of the digits 0–9 exclusively, and no number will have more than 5 digits.

Output

Output for each instance should consist of one line of either the form

```
Minimum base = x.
```

where x is the the smallest base ≤ 10 which results in an arithmetic sequence, or you should output

```
No base <= 10 can be found.
```

Sample Input

```
4
1 1010 2012 10021
3
11 33 55
4
11 33 55 77
5
10 160 340 520 1000
5
10 160 340 520 10000
0
```

Sample Output

```
Minimum base = 3.
Minimum base = 6.
Minimum base = 8.
Minimum base = 7.
No base <= 10 can be found.
```

Problem D: Untidy Desktops

Many companies are moving towards "paperless offices" in an attempt to increase productivity. Unfortunately, people who are unorganized with papers are unorganized without papers too! Instead of having piles of paper scattered all over a desk, people have windows overlapping and covering one another. For these people, finding the right window on the computer screen is just as hard as finding the right piece of paper on the desk.

For this problem, you will be given the locations and sizes of n windows on a computer desktop ($1 \leq n \leq 50$). You have been asked to evaluate how untidy the computer desktop is by counting the number of windows that overlap with at least one other window. Two windows overlap if at least one pixel is in both windows (including the boundary of the windows).

Input

The input will consist of multiple cases. Each case will start with a line containing a single integer n. This will be followed by n lines of the form

$$r\ c\ w\ h$$

where r, c are the row and column coordinates of the upper left corner of the window, and w, h are the width and height of the window. You may assume that the upper left corner of the computer desktop has coordinates (0,0), the screen has 1024 rows and 1280 columns, and all windows are completely within the boundary of the screen. A value of $n = 0$ will terminate the input.

Output

For each test case, print on a single line the number of windows that overlap with at least one other window.

Sample Input

```
3
0 0 20 20
20 20 20 20
40 40 20 20
5
0 0 20 20
18 18 20 20
40 40 20 20
5 10 4 2
100 100 40 40
0
```

Sample Output

```
0
3
```