

**University of Michigan-Dearborn  
2008 CIS Programming Contest  
December 3, 2008**

**Sponsored by ACM, SOAC:  
Association of Computing Machinery  
Student Organization Advisory Council**

Rules:

1. There are five questions to be completed in three hours.
2. 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.
3. The allowed programming languages are C, C++ and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
7. All communication with the judges will be handled by the PC2 environment.
8. Judges' decisions are to be considered final. No cheating will be tolerated.

## Problem A. Short On Cheese

Willey, the ratmouse, is fond of mazes. A ratmouse is a peculiar breed of lab rodents that love cheese more than rats and mice combined. Willey also likes mazes. Typically, when presented with a new maze, Willey likes to take his time and explore every twist and turn of the maze before getting to the cheese. Today, however, Willey is very hungry the evil lab guys have locked up all the cheese except the one in the mazes. Willey is smart enough to figure out that he needs to find the shortest way through the mazes to get his cheese fast, but he is too hungry to spend the time to figure out which path it is. One thing that can help Willey is knowing the length of the shortest path through the maze. Once known, Willey can use it to his advantage and avoid all paths that are longer than the shortest path. Mazes are set up in such a way that Willey can only move up, down, left or right. No diagonal moves are allowed. Since lab guys have set up way too many mazes to solve, Willey obviously needs your help. He wants you to write a program to find the length of the shortest path through the mazes, so he can gorge up on cheese and relax in his ratmouse house.

### Input

Input will consist of multiple test cases. First line will contain integer numbers  $r$  and  $c$ , where  $r$  is the number of rows in the maze to follow and  $c$  is the number of columns. The following input will contain  $r$  lines consisting of  $c$  characters each, where the character '#' depicts a wall of the maze and character '.' (dot) signifies a pathway within the maze. Willey will always start at the top left corner of the maze and end in the bottom right corner, where the lab guys hide the cheese. The values  $r$  and  $c$  will be within the intervals of  $1 \leq r \leq 100$ , and  $1 \leq c \leq 100$ . The last line will contain two zeroes and should not be processed.

### Output

For each test case, the output should contain a single integer on a line by itself representing the shortest length of Willey's path through the maze, starting at the top left corner and ending at the bottom right corner of the maze. You may assume that there will always be a path through the maze.

### Sample Input

```
4 7
..#....
#...##.
#.#.###.
#.....
2 3
..#
#..
0 0
```

### Sample Output

```
10
4
```

## Problem B. Pass or Feil

Professor Will Feil Yuhnnow is having a ball. He has discovered that his class has an unusual amount of students named Alex. To find out just exactly how many Alexes are in his class, he wrote down each student's name on a sheet of paper. He got the following list:

John, Alex, Michael, Alex, John, Alex, Peter, Eugene, Dave, David, Alex

Then, he counted up the number of Alexes he wrote down, and he was right! There were 4 students named Alex, more Alexes than students with any other name in the class. "Hmm", Professor Will thought to himself, "It will be quite interesting to find out the most popular student name in the entire school and how many students have that name". The longer he thought about this the more he wanted to know the answer. But then he would have to write down the entire list of students from the entire school. That's more than 2,000 people! Then he would have to figure out which name was the most popular and then count them up. The more he thought about it the more tedious it became. Professor Will realized that he fell short of his own expectations. Recalling that his class has a very bright student – you! – he now seeks your help. Professor wants you to write a program that takes a list of names, and figures out how many students on campus have the most popular name and what that name is. And if you come up with the answer, he will not fail you. If you don't come up with the answer you will have to put up with Prof. Feil's smugly-triumphal attitude for the rest of the semester. Clearly, you better get to work.

### Input

Input will consist of multiple cases. Each line will contain an integer  $n$  between 1 and 2,500 inclusively, where  $n$  the number of student names to follow, followed by  $n$  student names separated by a space. Each name will be no longer than 20 characters and consist of characters 'a' through 'z' with the first letter capitalized. Last line will start with 0, and should not be processed. You are guaranteed that there will be a unique most popular name for every case.

### Output

For each line of input professor expects you to print out the most popular name in the list, followed by the number of how many students have that name, with a space in between. The name should be capitalized as it was in the original list.

### Sample Input

```
11 John Alex Michael Alex John Alex Peter Eugene Dave David Alex
10 Mike Luke Cale Trevor Dennis Andrew Scott Garegin Bruce Mike
0
```

### Sample Output

```
Alex 4
Mike 2
```

## Problem C. Stuck In Traffic

You are stuck in traffic. Again. Instead of being bored you decide to observe what happens. You notice that most of the drivers are speeding up from 0 to few miles per hour and then they step on the brake to avoid hitting the car in front of them – your typical stop and go traffic. Since you have nothing else to do, you start thinking. What if instead of repeatedly accelerating and decelerating, you could compute the average speed of cruising, so that you could glide through the slow spot at one particular speed, without hitting your brakes even once. For simplicity you may assume that cars can speed up to any particular speed and stop instantaneously when brakes are pressed.

### Input

First line will start with an integer number  $n$  between 1 and 100, specifying the number of intervals to follow. The next  $n$  lines will each have two real numbers  $v$  and  $t$ , where  $v$ , between 0 and 200 inclusively, is the speed in mph that the car maintains during the time period of  $t$  seconds, and  $t$  is time in seconds between 1 and 10,000 inclusively. Each interval describes the speed of the car during stop and go traffic. A speed of 0 indicates that the car is not moving. There will be multiple test cases. The last line will contain the number 0 by itself and should not be processed.

### Output

Output will contain the average speed that will allow you to coast through the slow spot at one particular speed, without hitting the brakes or accelerating, while still getting you to your destination in the same amount of time had you chosen to follow stop and go traffic. Your output should be rounded to two decimal places by using conventional rounding rules. For example, 1.425 is rounded to 1.43, and 1.424 is rounded to 1.42.

### Sample Input

```
4
4 1
0 1
6 1
0 1
2
10 1
0 6
0
```

### Sample Output

```
2.50
1.43
```

## Problem D. Wordy Calculator

Your cousin asks you why computers don't understand numbers that are written out as words, such as "five" or "sixty six". You decide to write a program to show your cousin that computers can understand these numbers and even do operations on them.

### Input

Each line of input will start with an integer  $n$ , followed by a sentence starting with a capital letter and consisting of  $n$  words. The sentence will represent two numbers being added together. Each of the numbers in the sentence will be between 0 and 99 inclusively, written out as words with the word "plus" between them. Sentences will have no punctuation. The last line of input will contain the number 0 and should not be processed.

### Output

You are asked to perform the summation and produce the output as a written sentence with no punctuation and starting with a capital letter.

### Sample Input

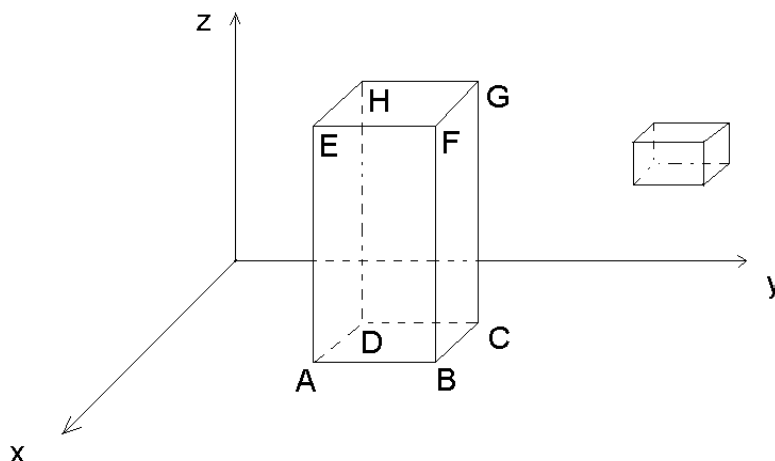
```
3 Seven plus nine
5 Eighty one plus seventy seven
3 Zero plus nine
4 Ninety nine plus sixty
0
```

### Sample Output

```
Sixteen
One hundred and fifty eight
Nine
One hundred and fifty nine
```

## Problem E. Think Inside the Box

Mikey is a bored and lonely student taking Calc III. One of his assignments features a 3-dimensional coordinate system with two 3-dimensional rectangular parallelepipeds. Mikey calls them 'boxes' to save on typing space. To make things more interesting for himself and to avoid boredom, Mikey decides to answer the age old question – is it possible to fit one of the boxes inside the other one. For example, in the graph below there are two boxes. The first box has dimensions of  $2 \times 3 \times 8$ , while the second box has dimensions of  $1 \times 3 \times 1$ . The second box fits inside the first one, but the first one does not fit inside the second one.



In order to fit inside, the inside box has to be strictly smaller than the outside box. In case the boxes are of identical sizes, Mikey considers them to be able to fit within each other. Mikey will consider rotating boxes to make them fit. So, for example, boxes with dimensions  $2 \times 3 \times 4$  and  $4 \times 3 \times 2$  are considered to be identical. After doing a few of these exercises Mikey is no longer bored. But he is still lonely. To cheer him up and to keep him company, will you help him with the remaining problems?

### Input

First line will contain an integer  $n$ , between 1 and 100 inclusively, signifying the number of test cases to follow. Each of the following  $n$  lines will contain a description of two boxes by listing a series of 16 points in  $(x, y, z)$  format. This makes for 48 numbers total per line. The first 3 numbers signify point A for box 1 with coordinates  $(x_A, y_A, z_A)$ , the next 3 numbers signify point B for box 1 with coordinates  $(x_B, y_B, z_B)$ , and so on. The first 8 points (24 numbers) belong to the first box, and the next 8 points belong to the second box. Each of the numbers given will be between -200 and 200 inclusively. You may assume that each of the box's sides will be parallel to the coordinate system axis. The order of the box's corners will always be given in order of points A, B, C, D, E, F, G, and H as in the diagram above.

### Output

For each test case, output a line in the following form: "Case  $k$ : box X fits inside box Y.", where  $k$  is the test case number starting with 1, and X and Y are either 1 or 2. If boxes are identical, output "Case  $k$ : boxes are identical." If neither box fits inside the other, output "Case  $k$ : neither box fits inside the

other.” You are also required to separate test cases with a blank line. No blank lines should be printed after the last test case or before the first test case.

### Sample Input

```
2
6 5 1 6 8 1 4 8 1 4 5 1 6 5 9 6 8 9 4 8 9 4 5 9 6 6 7 6 9 7 5 9 7 5 6
7 6 6 8 6 9 8 5 9 8 5 6 8
2 0 0 2 2 0 0 2 0 0 0 0 2 0 2 2 2 0 2 2 0 0 2 2 0 0 2 2 0 0 2 0 0 0
0 2 0 2 2 2 2 0 2 2 0 0 2
```

### Sample Output

Case 1: box 2 fits inside box 1.

Case 2: boxes are identical.