UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

II MARATON INTERNA DE PROGRAMACION

MARATON SEMANA TECNOLOGICA

CATEGORIA PRINCIPIANTES

UNIVERSIDAD DISTRITAL

31 de Octubre de 2007

PROBLEMAS

Elaborado por: Hector Florez
Basado de www.acis.org.co

Problem 1. Quicksum

Source file name: quicksum.c, quicksum.cpp or quicksum.java
Input: quicksum.in
Output: standar output

A checksum is an algorithm that scans a packet of data and returns a single number. The idea is that if the packet is changed, the checksum will also change, so checksums are often used for detecting transmission errors, validating document contents, and in many other situations where it is necessary to detect undesirable changes in data.

For this problem, you will implement a checksum algorithm called Quicksum. A Quicksum packet allows only uppercase letters and spaces. It always begins and ends with an uppercase letter. Otherwise, spaces and letters can occur in any combination, including consecutive spaces.

A Quicksum is the sum of the products of each character's position in the packet times the character's value. A space has a value of zero, while letters have a value equal to their position in the alphabet. So, A = 1, B = 2, etc., through Z = 26. Here are example Quicksum calculations for the packets "ACM" and "MID CENTRAL":

ACM: $1 \times 1 + 2 \times 3 + 3 \times 13 = 46$
MID CENTRAL: $1 \times 13 + 2 \times 9 + 3 \times 4 + 4 \times 0 + 5 \times 3 + 6 \times 5 + 7 \times 14 + 8 \times 20 + 9 \times 18 + 10 \times 1 + 11 \times 12 = 650$

Input

The input consists of one or more packets followed by a line containing only # that signals the end of the input. Each packet is on a line by itself, does not begin or end with a space, and contains from 1 to 255 characters.

Output

For each packet, output its Quicksum on a separate line in the output.

Sample input	Sample Output
ACM MID CENTRAL REGIONAL PROGRAMMING CONTEST ACN A C M ABC BBC #	46 650 4690 49 75 14 15

Problem 2. Strange Array Sum

Source file name: strange.c, strange.cpp or strange.java
Input: strange.in
Output: standar output

Given an Array of integers calculate the Strange Sum of this Array of numbers. The Strange Sum of the Array results of the sum of each of its numbers, and the addition of the maximum and minimum number in the array.

Input

There a several cases. Each case is giving by a line, it is containing a space separated list of integers representing the array. Each element of the array will be between 0 and 9.000.000, and the number of elements will never exceed 100.

Output

For each line in the input, the output contains the result of the Strange Sum.

Sample input	Sample Output
9 4 8 1 2 10 87 1 16 10 15 8 7 3 4	226 65

Problem 3. Contest System Quality Assurance Tester

Source file name: quality.c, quality.cpp or quality.java
Input: quality.in
Output: standard output

Write a program to score a small, three-problem programming contest. Each input line contains six space-separated integers representing raw score data. The first three integers are in the range 0 . . . 100000. They represent seconds taken to solve the first, second, and third problems, respectively. Zero seconds indicates that a problem has not been solved. The last three integers are in the range 0 . . . 100, representing the attempts taken to solve the first, second, and third problems, respectively. Every failed attempt is penalized with 20 minutes, but only for problems that are eventually solved.

Each output line should begin with the string team, followed by a single space, the input line number, a colon, a single space, the number of solved problems, a comma, a single space, and the total number of seconds including penalties it took for the solved problems.

Sample input	**Sample Output**
0 777 0 4 1 1 1 1 1 1 1 1	team 1: 1, 777 team 2: 3, 3

Problem 4. Factorials !!!

Source file name: factorials.c, factorials.cpp or factorials.java
Input: factorials.in
Output: standar output

Definition

n!!..! = n(n-k)(n-2k)..(n mod k), if k doesn't divide n
n!!..! = n(n-k)(n-2k)..k, if k divides n (There are k marks ! in the both cases).

For example, 10 mod 3 = 1; 3! = 3 x 2 x 1; 10!!! = 10 x 7 x 4 x 1;

Given numbers n and k we have calculated a value of the expression in the first definition. Can you do it as well?

Input

There will be multiple cases. Each test case will be contained on one line. Each line will start with the followed by exactly one space, then k exclamation marks.

Output

For each test case you should print one line of output which contains one number – n !!..! (there k marks ! here)

Sample input	Sample Output
3 ! 10 !!! 9 !!	6 280 945

Problem 5. Making Book

Source file name: book.c, book.cpp or book.java
Input: book.in
Output: standar output

A printer - who still uses moveable type - is preparing to print a set of pages for a book. These pages are to be numbered, as usual. The printer needs to know how many instances of each decimal digit will be required to set up the page numbers in the section of the book to be printed.

For example, if pages 10, 11, 12, 13, 14 and 15 are to be printed, computing the number of digits is relatively simple: just look at the page numbers that will appear, and count the number of times each digit appears. The digit 0 appears only once, the digit 1 appears 7 times, the digits 2, 3, 4 and 5 each appear once, and 6, 7, 8 and 9 don't appear at all.

Your task in this problem is to provide the printer with the appropriate counts of the digits. You will be given the numbers of the two pages that identify the section of the book to be printed. You may safely assume that all pages in that section are to be numbered, that no leading zeroes will be printed, that page numbers are positive, and that no page will have more than three digits in its page number.

Input

There will be multiple cases to consider. The input for each case has two integers, A and B, each of which is guaranteed to be positive. These identify the pages to be printed. That is, each integer P between A and B, including A and B, is to be printed. A single zero will follow the input for the last case.

Output

For each input case, display the case number (1, 2, . . .) and the number of occurrences of each decimal digit 0 through 9 in the specified range of page numbers. Display your results in the format shown in the examples below.

Sample input	Sample Output
10 15 912 912 900 999 0	Case 1: 0:1 1:7 2:1 3:1 4:1 5:1 6:0 7:0 8:0 9:0 Case 2: 0:0 1:1 2:1 3:0 4:0 5:0 6:0 7:0 8:0 9:1 Case 3: 0:20 1:20 2:20 3:20 4:20 5:20 6:20 7:20 8:20 9:120

Problem 6. Average

Source file name: average.c, average.cpp or average.java
Input: average.in
Output: standar output

Given an Array of integers calculate the average between all of them. The average is the result of the sum of each of its numbers divided by the quantity of numbers into the array.

Input

There a several cases. Each case is giving by a line, it is containing a space separated list of integers representing the array. Each element of the array will be between 0 and 100.000, and the number of elements will never exceed 100.

Output

For each line in the input, the output contains the result of the average.

Sample input	Sample Output
4 6 4 5 1 6 5 5 4 6 9 7	4 6