UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

MARATON SEMANA TECNOLOGICA

CATEGORIA AVANZADOS

UNIVERSIDAD DISTRITAL

27 de Octubre de 2010

PROBLEMAS

Elaborado por: Hector Florez

Problem 1.

Hello Kitty

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

Kitty sends a kind of original email messages to her friend Garf. To write a message, she chooses a word W and a number n and replicates W n times horizontally. Then she repeats this string in the next line, but rotating the characters once to the left. And she repeats this 'rotateand-output' process until the word W appears displayed as the first column of the rectangular pattern that she produces.

As an example, when she chooses the word Hello and the number 3, she gets the pattern:

```
HelloHelloHello
elloHelloHelloH
lloHelloHelloHe
loHelloHelloHel
oHelloHelloHell
```

Kitty has been sending such emails during the last three years. Recently, Garf told her that perhaps her work may be automatized with a software to produce Kitty's patterns. Could you help her?

Input

The input file contains several test cases, each one of them in a separate line. Each test case has a word and a positive integer that should generate the corresponding rectangular pattern. The word is a string of alphabetic characters (a..z). The number is less than 10. A line whose contents is a single period character means the end of the input (this last line is not to be processed).

Output

Output texts for each input case are presented in the same order that input is read. For each test case the answer must be a left aligned Kitty pattern corresponding to the input.

Sample input	Sample Output
Love 1 Kitty 2 .	Love oveL veLo eLov KittyKitty ittyKittyK ttyKittyKi tyKittyKit yKittyKity

Problem 2.

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, 14and15 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 3

Electrical Outlets

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

Roy has just moved into a new apartment. Well, actually the apartment itself is not very new, even dating back to the days before people had electricity in their houses. Because of this, Roy's apartment has only one single wall outlet, so Roy can only power one of his electrical appliances at a time.

Roy likes to watch TV as he works on his computer, and to listen to his HiFi system (on high volume) while he vacuums, so using just the single outlet is not an option. Actually, he wants to have all his appliances connected to a powered outlet, all the time. The answer, of course, is power strips, and Roy has some old ones that he used in his old apartment. However, that apartment had many more wall outlets, so he is not sure whether his power strips will provide him with enough outlets now.

Your task is to help Roy compute how many appliances he can provide with electricity, given a set of power strips. Note that without any power strips, Roy can power one single appliance through the wall outlet. Also, remember that a power strip has to be powered itself to be of any use.

Input

Input will start with a single integer 1 <= N <= 20, indicating the number of test cases to follow. Then follow N lines, each describing a test case. Each test case starts with an integer 1 <= K <= 10, indicating the number of power strips in the test case. Then follow, on the same line, K integers separated by single spaces, O1 O2 ... O_k, where 2 <= O_i <= 10, indicating the number of outlets in each power strip.

Output

Output one line per test case, with the maximum number of appliances that can be powered.

Sample input	Sample Output
3 3 2 3 4 10 4 4 4 4 4 4 4 4 4 4 4 10 10 10 10	7 31 37

Problem 4.

Rafting

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

A whitewater rafting company is trying to fit its clients into as few rafts as possible. Each raft can take one or two people and has a weight limit. You will be given the weight limit for rafts and the list of clients' weights. Compute the minimum number of rafts needed to accommodate all clients.

Input

The first line of input contains the number of test cases. The test cases follow. Each test case consists of two lines. The first line contains two integers n and w, 1 <= n; w <= 1000. n is the number of clients, w is the weight limit of each raft. The second line contains n integers between 1 and 1000, the weights of the clients.

Output

For each test case output a single line to the standard output. It should contain the minimum number of rafts or the word IMPOSSIBLE if no assignment is possible.

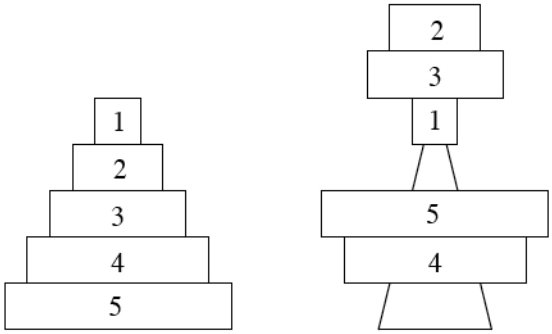
Sample input	Sample Output
2 3 100 95 13 25 3 100 1000 1000 1000	2 IMPOSSIBLE

Problem 5.

Stacking Tower

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

One of the most common children's toys is a stacking tower, which consists of a series of rings of different sizes and a tapered rod which can hold the rings. The rings and rod are designed so that when the rings are placed in descending order by size, they fit exactly on the rod. Further, each ring, if placed by itself on the rod will go no lower than its position when all rings are on the rod. The diagram on the left shows an example of this toy which uses five rings, numbered 1 (smallest) to 5 (largest)



Unless endowed with super–genius mental faculties, most infants will place the rings in a random order on the rod, which results in some rings sticking over the top of the rod. The above diagram on the right shows the result of one such random placement. Your job will be to determine the number of rings which sit above the rod given a random ordering of the rings. You may assume that the rings may be stacked arbitrarily high without falling over.

Input

Input will consist of multiple problem instances. Each instance consists of a single line of the form n r_1 r_2 … r_n. The positive integer n specifies the number of rings (<= 100), and the remaining n integers give the order the rings are put on the rod. A value of n = 0 will terminate input.

Output

For each problem instance, you should output one line containing an integer indicating the number of rings sticking over the top of the rod.

Sample input	Sample Output
5 5 4 3 2 1 5 4 5 1 3 2 8 1 2 3 4 5 6 7 8 20 11 10 19 7 12 14 5 2 3 1 8 6 13 17 18 9 4 20 15 16 0	0 2 7 11

Problem 6

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