ACM International Collegiate Programming Contest 2011-2012

Sponsored by IBM

Qualification Local Contest

for SWERC 2011

University of Valladolid, Spain

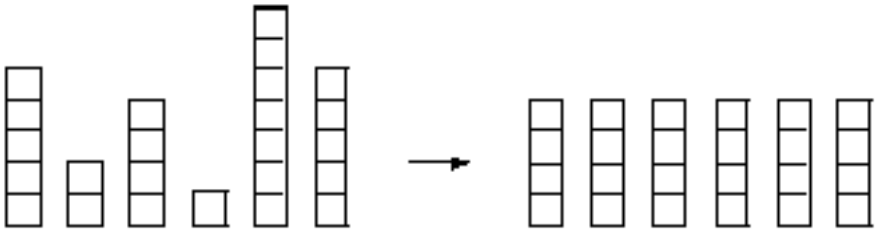
October 21st, 2011

Problem Set

This problem set should contain six (6) problems on eleven (11) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Problem A: Box of Bricks

Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. ``Look, I've built a wall!", he tells his older sister Alice. ``Nah, you should make all stacks the same height. Then you would have a real wall.", she retorts. After a little con- sideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?



Input

The input consists of several data sets. Each set begins with a line containing the number n of stacks Bob has built. The next line contains n numbers, the heights h_i of the n stacks.

You may assume $1 \leq n \leq 50$ and $1 \leq h_i \leq 100$.

The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height.

The input is terminated by a set starting with $n = 0$. This set should not be processed.

Output

For each set, first print the number of the set, as shown in the sample output. Then print the line ```The minimum number of moves is` k.", where k is the minimum number of bricks that have to be moved in order to make all the stacks the same height.

Output a blank line after each set.

Sample Input

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

Sample Output

```
Set #1
The minimum number of moves is 5.
```

Problem B: Peter's Smokes

Peter has **n** cigarettes. He smokes them one by one keeping all the butts. Out of **k** > **1** butts he can roll a new cigarette.

How many cigarettes can Peter have?

Input

Input is a sequence of lines. Each line contains two integer numbers giving the values of **n** and **k**. The input is terminated by end of file.

Output

For each line of input, output one integer number on a separate line giving the maximum number of cigarettes that Peter can have.

Sample Input

```
4 3
10 3
100 5
```

Sample Output

```
5
14
124
```

Source: University of Alberta Local Contest

Problem C: What's Cryptanalysis?

Cryptanalysis is the process of breaking someone else's cryptographic writing. This sometimes involves some kind of statistical analysis of a passage of (encrypted) text. Your task is to write a program which performs a simple analysis of a given text.

Input

The first line of input contains a single positive decimal integer *n*. This is the number of lines which follow in the input. The next *n* lines will contain zero or more characters (possibly including whitespace). This is the text which must be analyzed.

Output

Each line of output contains a single uppercase letter, followed by a single space, then followed by a positive decimal integer. The integer indicates how many times the corresponding letter appears in the input text. Upper and lower case letters in the input are to be considered the same. No other characters must be counted. The output must be sorted in descending count order; that is, the most frequent letter is on the first output line, and the last line of output indicates the least frequent letter. If two letters have the same frequency, then the letter which comes first in the alphabet must appear first in the output. If a letter does not appear in the text, then that letter must not appear in the output.

Sample Input

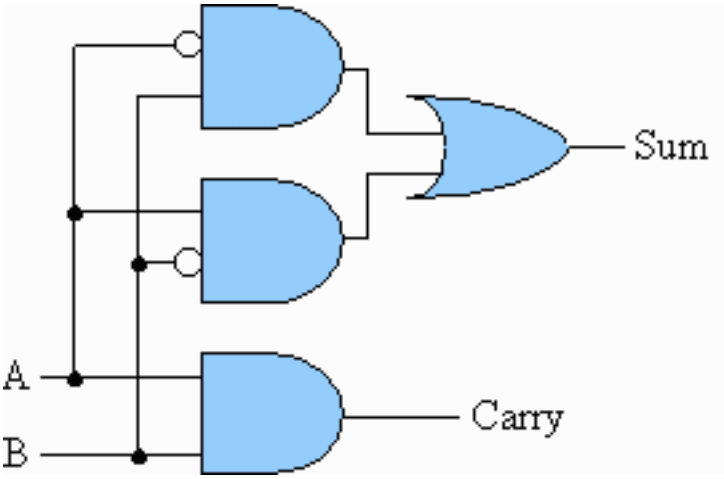
```
3
This is a test.
Count me 1 2 3 4 5.
Wow!!!!  Is this question easy?
```

Sample Output

```
S 7
T 6
I 5
E 4
O 3
```

A 2
H 2
N 2
U 2
W 2
C 1
M 1
Q 1
Y 1

Problem D: To Carry or not to Carry



6+9=15 seems okay. But how come 4 +6=2?

You see, Mofiz had worked hard throughout his digital logic course, but when he was asked to implement a 32 bit adder for the laboratory exam, he did some mistake in the design part. After tracing the design for half an hour, he found his flaw!! He was doing bitwise addition but his carry bit always had zero output. Thus,

```
 4 = 00000000 00000000 00000000 00000100
+6 = 00000000 00000000 00000000 00000110
-----------------------------------------
 2 = 00000000 00000000 00000000 00000010
```

Its a good thing that he finally found his mistake, but it was too late. Considering his effort throughout the course, the instructor gave him one more chance. Mofiz has to write an efficient program that would take **2 unsigned 32 bit decimal numbers** as input, and produce an **unsigned 32 bit decimal number** as the output adding in the same was as his circuit does.

Input

In each line of input there will be a pair of integer separated by a single space. Input ends at EOF.

Output

For each line of input, output one line -- the value after adding the two numbers in the "Mofiz way".

Sample Input

```
4 6
6 9
```

Sample Output

```
2
15
```

Problem setter: Monirul Hasan (Tomal), CSE Dept, Southeast University, Bangladesh

Problem E: Rational Numbers from Repeating Fractions

A rational number is any which can be written in the form *p*/*q*, where *p* and *q* are integers. All rational numbers less than 1 (that is, those for which *p* is less than *q*) can be expanded into a decimal fraction, but this expansion may require repetition of some number of trailing digits. For example, the rational number 7/22 has the decimal expansion .3181818... Note that the pair of digits 1 and 8 repeat ad infinitum. Numbers with such repeating decimal expansions are usually written with a horizontal bar over the repeated digits, like this: $.3\overline{18}$

If we are given the decimal expansion of a rational fraction (with an indication of which digits are repeated, if necessary), we can determine the rational fraction (that is, the integer values of *p* and *q* in *p*/*q*) using the following algorithm.

Assume there are *k* digits immediately after the decimal point that are not repeated, followed by a group of *j* digits which must be repeated. Thus for 7/22 we would have $k = 1$ (for the digit 3) and $j = 2$ (for the digits 1 and 8). Now if we let *X* be the original number (7/22), we can compute the numerator and denominator of the expression

$$\frac{10^{k+j} \times X - 10^{k} \times X}{10^{k+j} - 10^{k}}$$

For $.3\overline{18}$ we obtain the following calculation for the numerator of this fraction:

$$10^{3} \times .3\overline{18} - 10^{1} \times .3\overline{18} = 318.\overline{18} - 3.\overline{18} = 315$$

The denominator is just 1000 - 10, or 990. It is important to note that the expression in the numerator and the denominator of this expression will always yield integer values, and these represent the numerator and denominator of the rational number. Thus the repeated fraction $.3\overline{18}$ is the decimal expansion of the rational number 315/990. Properly reduced, this fraction is (as expected) just 7/22.

Input

The input data for this problem will be a sequence of test cases, each test case appearing on a line by itself, followed by a `-1`. Each test case will begin with an integer giving the value of *j*, one or more spaces, then the decimal expansion of a fraction given in the form *0.ddddd* (where *d* represents a decimal digit). There may be as many as nine (9) digits in the decimal expansion (that is, the value of *k*+*j* may be as large as 9).

Output

For each test case, display the case number (they are numbered sequentially starting with 1) and the resulting rational number in the form *p*/*q*, properly reduced.

Sample Input

```
2 0.318
1 0.3
2 0.09
6 0.714285
-1
```

Sample Output

```
Case 1: 7/22
Case 2: 1/3
Case 3: 1/11
Case 4: 5/7
```

Problem F: One Little, Two Little, Three Little Endians

Writing programs that are completely portable across different operating systems, operating system versions and hardware platforms is a challenging task. One of the difficulties encountered is a result of decisions made by hardware manufacturers about how they will store integer data in memory. Because these representations can differ from machine to machine, sharing binary data often cannot be done without modifying the way in which the data is stored or the way in which it is handled by one or more of the platforms.

Fortunately there is near-universal agreement among hardware manufacturers that addressable memory be ordered into 8-bit bytes. For integer data values that require more than 8-bits, such as the typical 2- byte, 4-byte, and 8-byte integer types available on most modern hardware, there is no such agreement and two incompatible storage schemes exist. The first stores integers as groups of consecutive 8-bit bytes with the least significant byte occupying the lowest memory location within the group and the most significant byte occupying the highest memory location. The second is just the reverse; the least significant byte is stored in the highest memory location within the group, and the most significant byte is stored in the lowest memory location. The computing industry has dubbed these schemes Little Endian and Big Endian, respectively. There is also near-universal agreement that signed integers are stored using "two's complement" representation, and you may assume that this is the case.

When binary integer data is shared between a Little Endian and Big Endian machine, a data conversion must be performed which involves reversing the bytes of the data. Once the bytes have been reversed the integer is then correctly interpreted by the hardware as the original value from the opposite-endian machine. The object of this problem is to write a program that will read a list of integers and report the integers that the binary representations of the input integers would represent on an opposite-endian machine.

Input

The input will consist of a list integers. The end of the input file marks the end of the list. All input integers can be represented as a 32-bit signed integer value. That is, the input integers will be in the range -2147483648 to 2147483647.

Output

For each input integer a single line should be printed to the output file. The line should

contain the input integer followed by the phrase ``converts to'' followed by one space followed the other-endian value.

Sample Input

```
123456789
-123456789
1
16777216
20034556
```

Sample Output

```
123456789 converts to 365779719
-123456789 converts to -349002504
1 converts to 16777216
16777216 converts to 1
20034556 converts to -55365375
```