Computers lose value very fast. Just when you thought you had the latest and best, two weeks later another model is announced and your computer is not nearly as valuable!

Write a program which will compute just how much less valuable your computer is after a period of weeks, assuming a constant depreciation facter. You will be given the original value of the computer, and the percent by which it loses value ("depreciates") each week. For example, if the original value was $5000.00 and the rate of depreciation is 2%, then after one week it has lost $100.00, so its value is $4900.00. Next week it loses 2% of $4900, i.e. $98.00, so its value is $4802.00. Next week it loses $96.04, so its value is $4705.96. The next week it loses $94.1192, but because we can't deal in fractions of a cent, this is trancated to $94.11, so its value after four weeks is $4611.85. Note that this is not the same as you would get by simply taking 4 x 2% off $5000 - this would give $4600.00.

Input will consist of a real number and two integers - the original value of the item in dollars and cents, the depreciation per time period as a percentage, and the number of time periods. The original value will always be no greater than $100,000.00. The percentage depreciation will always be an integer less than 100 and greater than 0. The number of time periods will always be no greater than 500.

Output will be one number giving the final value of the item, in dollars and cents (i.e. with two decimal places). Note that the amount depreciated must be computed for each period and then truncated to give whole cents, as shown in the calculation above.

In the following examples, please note that each line represents a separate file (input stream) and therefore a separate execution of the program.

`5000.00 2 4`	`4611.85`
`44281.15 10 10`	`15439.91`

One of the stated aims of the so-called 'seamless' education system is that students' academic records follow them from one institution to another. This sounds fine except that the formats, the number of fields and even the characters used to separate fields can all differ. Thus one institution may have 6 fields separated by commas while another may have 8 fields separated by tabs. In the general case this can be a very difficult problem, but for this situation we will simplify it considerably.

Write a program that will read a line containing fields separated by commas. There should be 6 fields in each line and hence 5 commas, no more and no less. Some fields may be blank; we wish to know how many of them. You may assume that no line contains more than 60 characters.

Determine how many commas the line contains. If it does not contain exactly 5 commas write 'Invalid', otherwise determine how many empty fields there are and write this number.

In the following example, please note that each line represents a separate file (input stream) and therefore a separate execution of the program.

```
Name,Address1,Address2,Address3,City,Country
Name,Address1,Address2,City,Country
Name,Address1,Address2,,City,Country
Name,Address1,Address2, ,City,Country
```

```
0
Invalid
1
1
```

Write a program that will read two words (separated by a single space), and determine the order that they would be printed in a dictionary. Although the "words" may contain non-alphabetic characters, only alphabetic characters should be considered when determining the order. Also, you should treat upper and lower case letters as being equivalent.

Input consists of a line containing two strings. Each string is a sequence of not more than 20 non-space characters. The strings are separated by exactly one space.

Output will consist of the first string, a space, one of the phrases 'is less than', 'is equal to' or 'is greater than' as appropriate, a space and the second string.

In the following examples, please note that each line represents a separate file (input stream) and therefore a separate execution of the program.

```
Aardvark elephant
elephant Aardvark
elephant Zebra
Eland elephant
**thiswillFOXthem! ThiswillfoxTHEM.
1st 21st
```

```
Aardvark is less than elephant
elephant is greater than Aardvark
elephant is less than Zebra
Eland is less than elephant
**thiswillFOXthem! is equal to ThiswillfoxTHEM.
1st is equal to 21st
```

Until early this year, standard New Zealand registration plates (number plates) consisted of two letters and up to 4 digits. These were apparently issued in sequence, i.e. every plate from AA1 through to ZZ9999 was issued. I say "apparently" because I have never actually seen a vehicle with a number 'next to' mine - close admittedly, but never next to.

This then raises the problem of what we mean by "close". Which of these is closest to NZ3868: NZ3878? NZ3870? NY3868? MZ3868? What about NA3868?

For the purposes of this problem we will define 'closeness' to be the overall numeric similarity. To determine this we count the number of differences and form their absolute sum. The first criterion is the number of differences (fewer is closer); if these are equal then the smaller sum determines the closeness. The examples should make this clear. The examples should make this clear. The differences are between the characters as they stand, without wraparound, so the difference between 'A' and 'Z' is 25 not 1.

Input will consist of three strings representing old style New Zealand regstration plates, i.e. two upper case letters followed by 4 digits. These strings will be separated by exactly one space. Your task is to determine which of the second two strings is closest (under the above definitions) to the first.

Output will consist of the closest plate to the first one. If both are equally close, print both.

In the following examples, please note that each line represents a separate file (input stream) and therefore a separate execution of the program.

```
NZ3868 NZ3768 NZ3859
NZ3868 NY3869 NZ3969
NZ3868 NZ3969 NZ3669
AA0001 AA0002 AB0001
```

```
NZ3768
NY3869 NZ3969
NZ3969
AA0002 AB0001
```