problem 4

Stellar Classification

5 points

JAVA program name must be prob04.java
C /C++ program name must be: prob04.exe

Introduction

When ordinary people use the word "color" to describe a star, they mean "what is the tint perceived by the eye?" One star might have a color of "pale orange", another "bluish-white". But astronomers often use the word "color" to refer to a star's ***Spectral Class***. The basic spectral classes are O, B, A, F, G, K, and M, where class O is the bluest and class M is the reddest. The spectral class can be determined from the ***B-V Color Index***, which is the difference between B and V magnitudes as measured in the UBV system:

$$\text{B-V Color Index} = m_B - m_V$$

Values for m_B and m_V are measured using passband filters. The ***B band*** covers a range of blue frequencies and the ***V band*** covers a range of green and yellow frequencies.

Write a program to classify star colors from raw B and V magnitude values.

Sample Input

Each line of input contains the name of a star, followed by two floating point values: m_B and m_V, respectively. The input ends with the word END.

Spectral Class	*B-V min value*	*B-V max value*
O	-0.350	-0.251
B	-0.250	-0.001
A	0.000	0.249
F	0.250	0.499
G	0.500	0.999
K	1.000	1.499
M	1.500	2.000

```
VEGA 0.00 0.00
PROCYON 0.80 0.38
SOL 5.48 4.83
ANTARES 2.96 1.09
POLLUX 2.14 1.14
CAPELLA 0.88 0.08
END
```

Sample Output

For each star the program must print the name of the star, the B-V color index, and the corresponding spectral class.

```
VEGA 0.00 A
PROCYON 0.42 F
SOL 0.65 G
ANTARES 1.87 M
POLLUX 1.00 K
CAPELLA 0.80 G
```

Problem 7

Ponzi Calculator

5 points

JAVA: program name must be prob07.java
C /C++ program name must be: prob07.exe

Task Description

You've decided to retire early using a fool-proof pyramid, or "Ponzi" scheme. A Ponzi scheme is a fraudulent investment proposal that uses money collected from new investors (victims) to pay off previous investors. The problem with a Ponzi scheme is that eventually, there just aren't enough new investors to pay off all the existing ones, and the pyramid collapses.

For your scheme, you're going to tell investors about the wonders of Emu farming. You promise to raise the Emus at a "secret farm in Montana", and sell their wool to high-end clothing makers. You'll pay them interest every month and investors will quickly double their money or more!

You need $10,000,000 to buy a beach house in the Caymen Islands. Write a program to figure out how much to steal, err, borrow, from each investor, how many investors each month you can swindle, and what interest rate will intice them to invest. When you have enough cash, you'll skip town. Simple!

The Annual Interest Rate will be a whole integer between 1-20. For simplicity, assume that each month is exactly 1/12 of a year, all new investors come in at the beginning of the month, and all interest payments are made at the end of the month.

Input

Prompt for the dollar amount each "valued investor" will contribute (1-1000000), the number of new investors you'll recruit each month (1-20000), and the Annual Interest Rate (1-400) you'll promise everyone. For simplicity, assume that each month is exactly 1/12 of a year, all new investors come in at the beginning of the month, and all payments are made at the end of the month.

Output

The program will output the balance sheet at the end of the month. For each month, print the number of investors, the account balance once once new investments are made, the interest to be paid at the end of the month, and the balance at the end of the month. If the balance before paying interest is over $10,000,000, print the total balance, the number of months that have passed, and the words "Purple Horseshoe loves Emu Farms!" - your secret phrase to let you know it's time to run for the airport. If at any time the balance goes below zero, the gig is up, so print out "Abort Project Emu Farms!".

Sample Inputs/Outputs

```
Investment Cost (per person) $:  5000
New Investors (per month):  500
Guaranteed Annual Interest Rate (%):  180

Month Investors Begin Balance  Interest Paid  End Balance
----- --------- -------------- -------------- ---------------
1          500     $2500000    $375000            $2125000
2          1000    $4625000    $750000            $3875000
3          1500    $6375000    $1125000           $5250000
4          2000    $7750000    $1500000           $6250000
5          2500    $8750000    $1875000           $6875000
6          3000    $9375000    $2250000           $7125000
7          3500    $9625000    $2625000           $7000000
8          4000    $9500000    $3000000           $6500000
9          4500    $9000000    $3375000           $5625000
10         5000    $8125000    $3750000           $4375000
11         5500    $6875000    $4125000           $2750000
12         6000    $5250000    $4500000           $750000
13         6500    $3250000    $4875000           -$1625000

Abort Project Emu Farms!
```

```
Investment Cost (per person) $:  5000
New Investors (per month):  500
Guaranteed Annual Interest Rate (%):  150

Month Investors Begin Balance  Interest Paid  End Balance
----- --------- -------------- -------------- ---------------
1          500     $2500000    $312500            $2,187500
2          1000    $4687500    $625000            $4,062500
3          1500    $6562500    $937500            $5,625000
4          2000    $8125000    $1250000           $6,875000
5          2500    $9375000    $1562500           $7,812500
6          3000    $10312500   $1875000           $8,437500

Total Balance at beginning of Month 6 is $10312500
Purple Horseshow loves Emu Farms!
```

problem **2**

Divide by 11

3 points

JAVA program name must be prob02.java
C /C++ program name must be: prob02.exe

Introduction

In honor of 2011 – your task is to accept a positive integer as input and, using the below algorithm, determine if that integer is divisible by 11. This algorithm was originally devised in 1897 by Charles Dodgson, a.k.a. Lewis Carroll.

The algorithm:

> As long as the number under test still has at least two digits, form a new number by:
> - deleting the ones digit
> - subtracting the deleted digit from the remaining, shortened number
>
> If the remaining two-digit number is divisible by 11, then the original number is divisible by 11.

HINT: You may need to use a wide integer data type, such as "long" in Java.

Sample Input

```
161408196180
```

Sample Output

```
161408196180
16140819618
1614081953
161408192
16140817
1614074
161403
16137
1606
154
11

The number 161408196180 is divisible by 11.
```

Return of Spell Binder

4 points

2012

Introduction

After his last defeat at the hands of Letterman, Spell Binder was sentenced to a year of prison time for vandalism of public words. During his time away he only grew more determined to sow his own peculiar brand of havoc. Spell Binder has created an army of speelbots whose dastardly purpose is to replace letters in words being used in public places. Letterman discovered one of these speelbots at Kelly's Custard stand, where the bot changed the CUSTARD into MUSTARD. That doesn't taste very nice at all, precious. Fortunately for Kelly and all of her customers, Letterman was able to change the M back into a C. However, Letterman cannot fight an entire army of speelbots on his own, nor does he have any computer skills to write his own bot.

Write a program that can undo the havoc caused by Spell Binder's speelbots.

Sample Input

The input will consist of a single word followed by two groups of one or more letters, all separated by spaces.

Example 1
```
MUSTARD M C
```

Example 2
```
JUNK J TR
```

Example 3
```
MONSTER ON A
```

Sample Output

The program must correct the input word by replacing the letters of the first group with the letters of the second group and print the corrected word.

Example 1
```
CUSTARD
```

Example 2
```
TRUNK
```

Example 3
```
MASTER
```

problem **3**
Weight Calculator
3 points

hp codewars 2013

Introduction
Did you know that if you were standing on the moon you'd only weigh about 1/6 of your weight on Earth? The surface gravity of celestial bodies (moons, planets, Pluto, etc.) in the solar system varies widely according to the body's mass and radius.

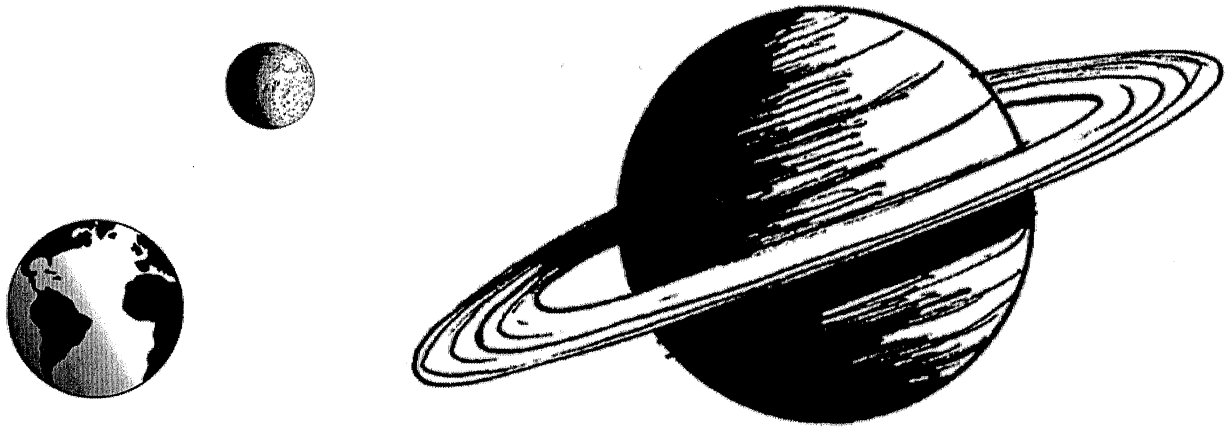
Write a program to compute a person's weight on the surface of a celestial body.

Input
Each line of input will contain a person's name, their weight (in pounds, on Earth), a single-word name of a celestial body, and a conversion factor for the surface gravity on that body. The last line of input is the word "END" followed by three zeros.

```
Fred 179.0 Luna 0.1654
Layla 131 Mars 0.376
Pat 145.2 Neptune 1.14
Rajavel 156.4 Ganymede 0.146
END 0 0 0
```

Output
The program must convert each weight and print the result using the format shown below. Weights must match the expected values within +/- 1 pound.

```
On Luna, Fred would weigh 29.6066 pounds.
On Mars, Layla would weigh 49.256 pounds.
On Neptune, Pat would weigh 165.528 pounds.
On Ganymede, Rajavel would weigh 22.8344 pounds.
```



Tribonacci Numbers

3 points

Introduction

You may recall that Fibonacci numbers are formed by a sequence starting with 0 and 1 where each succeeding number is the sum of the two preceding numbers; that is, F[n] = F[n-1] + F[n-2] with F[0] = 0 and F[1] = 1.

Tribonacci numbers are like Fibonacci numbers except that the starting sequence is 0, 1 and 1 and each succeeding number is the sum of the three preceding numbers; that is, T[n] = T[n-1] + T[n-2] + T[n-3] with T[0] = 0, T[1] = 1 and T[2] = 1.

The first eleven terms of the Tribonacci sequence are 0, 1, 1, 2, 4, 7, 13, 24, 44, 81 and 149.

Input

Each line of input is an integer. The maximum possible input value is 30. The last line of input is a -1.

```
3
9
11
0
-1
```

Output

For each non-negative input, the program must use the integer as an index to the Tribonacci sequence and print the Tribonacci number corresponding to that index.

```
2
81
274
0
```

Triangle Area

4 points

Introduction

One formula for the area of a triangle is:

$$\text{Area} = \frac{a \times b \times sin\, w}{2}$$

Where *a* and *b* are the lengths of two sides and *w* is the angle between those same two sides. The angle *w* can be computed in two steps, first by calculating *cos w* using the length *c* of the third side in this formula:

$$cos\, w = \frac{a^2 + b^2 - c^2}{2 \times \mathrm{a} \times \mathrm{b}}$$

Then use your language's arc-cosine function to compute *w*.

Java:	Math.acos()
C++:	acos()
Python:	math.acos()
JavaScript:	Math.acos()

The formula for the distance between two points is

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

Input

Each line of input will contain the x and y coordinates (in that order) for three distinct points. The input ends with six zeros.

```
3.1415 2.7777 -3.9123 0.2133 0.4324 -11.111
-8.675309 1.41421 9.999 0.0001 9.999 1.41421
0.7071 7.732 2.718 -1.005 -6.931 0.866025
0.6125 0.03125 99.999 0.9125 99.999 -0.56875
0 0 0 0 0 0
```

Output

For each line the program must print the area of the triangle. Answers should be accurate to within ±1 of the expected value. For example, if the expected value is 13.2038, any answer between 12.2038 and 14.2038 will be considered correct.

```
45.5104
13.2038
40.2704
73.6081
```

A Question of Time

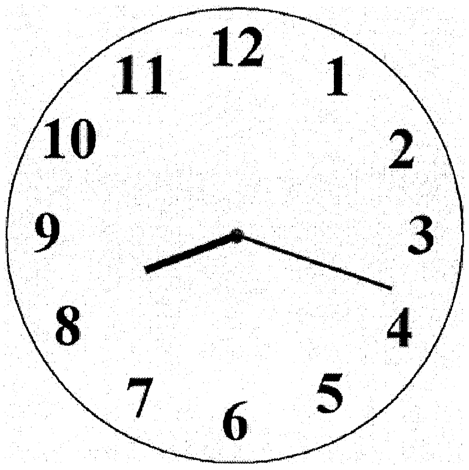
4 points

hp codewars 2013

Introduction

This program will calculate time based on a traditional round clock with hour and minute hands. The hour hand on this clock moves slowly from hour to hour; it does not jump from one hour to the next.

If both hands are on opposite sides of, and an equal distance from, the number six, then for a given number of minutes after the hour the program should determine the current time.



Input

Each line of input is an integer value indicating the number of minutes after the hour, in the range zero through fifty-nine. The input ends with a -1.

```
18
5
46
0
-1
```

Output

For each input, the program should print the time formatted digitally, i.e., "hours:minutes". Single-digit hours must not have a leading zero, but single-digit minutes must have a leading zero.

```
8:18
11:05
2:46
12:00
```

problem 1
Bee Math
2 points

Your team crawls through a short tunnel into an enormous beehive and is forced to wear protective clothing to pass through a swarm of bees. At the far end of the hive, you find a formula. It reads:

The population of this bee colony can be approximated by the equation

*P(t) = 100 * sqrt(t) + 201/(t+1) + 1*

where t is measured in days after the colonization of the beehive.

Write a program to compute the population of the beehive after a specified number of days.

Input

Each line of input is a positive integer value for t, the number of days after the colonization of the hive. The input ends with a zero.

```
7
38
24
0
```

Output

For each value of t, the program must print t and the population of the hive for that day, rounded to the nearest integer. Your result must match the expected value within +/- 1.

```
7 291
38 623
24 499
```

problem **2**
Check Digit
3 points

As you stroll by the arena gift shop, you luckily spot another event waiting for you. At the checkout register, the clerk is busy scanning UPC codes. Any product you buy has one of these. It helps identify the product being sold so that the store can manage inventory, pricing and other data. Universal Product Codes (UPC) are easy to spot by their bar code.

0 12345 67890 5

A UPC is a 12 digit number that encodes product information such as the manufacturer, product type, weight, and other data. The last digit in this number is the check digit. This extra digit helps verify that a tired programmer didn't get one digit wrong, or transpose a pair of numbers (ex. 34 -> 43) or otherwise alter the sequence.

Check Digit:
A check digit is used to ensure that a sequence of numbers was transmitted or entered correctly without human error. The algorithm used to calculate the check digit determines the types of errors it will catch. For UPC the algorithm catches 100% of single digit errors and 89% of transposition errors.

Your task is to calculate the missing check digit for the given list of UPCs.

Here's the UPC check digit algorithm:
- First, add all the digits in the odd-numbered positions together and multiply the result by three.
- Then, add the digits in the even-numbered positions to the result.
- Next, find modulo 10 of the sum. Modulo calculates the remainder after dividing the sum by 10.
- Finally, if the remainder is not zero, subtract it from 10.

Input
The first line of input contains the number of UPCs that follow. The digits of each UPC will be separated by one space.

```
6
0 1 2 3 4 5 6 7 8 9 0
0 3 6 0 0 0 2 9 1 4 5
0 7 3 8 5 2 0 0 9 3 8
0 7 0 7 3 4 0 5 3 1 6
0 4 1 2 2 0 1 8 9 0 4
0 3 7 0 0 0 2 0 2 1 4
```

Output
For each UPC, the program must print the UPC including the calculated check digit. The digits of each UPC should be separated by a single space.

```
0 1 2 3 4 5 6 7 8 9 0 5
0 3 6 0 0 0 2 9 1 4 5 2
0 7 3 8 5 2 0 0 9 3 8 5
0 7 0 7 3 4 0 5 3 1 6 0
0 4 1 2 2 0 1 8 9 0 4 5
0 3 7 0 0 0 2 0 2 1 4 1
```

Problem 4 Net OS App [3 points]

Summary

The IQ Computer Corporation* has recently acquired Hand Computing* and plans to release a new line of products based on Hand's innovative technology. IQ has rebranded HandOS with the name netOS and is preparing for product launch of its newest devices.

There's just one catch: netOS doesn't have nearly as many apps as those other guys.

Never ones to worry, the IQ marketing team has decided to offer free netOS devices to the first 1,000 software developers who submit apps for the new platform. Being a savvy software developer, you have decided to write a simple app that calculates the area of a room. The user just enters two integers for the length and width of a room and the app will display the area.

If your conscience starts to bother you because you're writing such a simplistic app, don't fret. As it turns out, IQ Computer Corporation has no intention of actually publishing any of the apps submitted, so writing a useful app would just be a waste of everyone's time. This is how multinational corporations roll.

Besides, it's not like 1,000 more apps would make a difference anyway.

Input

Each line of the input is two integer values. The last line of input is two zeros.

```
15 9
11 8
0 0
```

Output

For each line of input, the program must multiply the two input numbers and print the result.

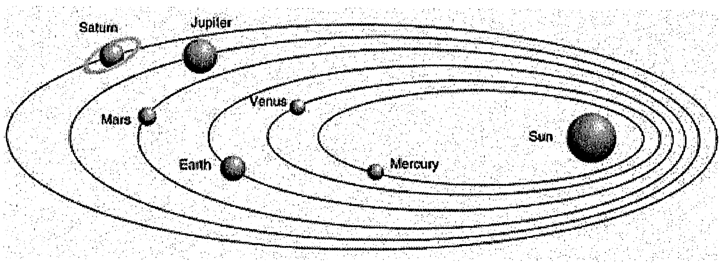
```
135
88
```

* NOTE-- The corporations appearing in this work are fictitious. Any resemblance to real corporations, profitable or bankrupt, is purely coincidental.

Problem 7 Kepler's Third Law [4 points]

Summary

Johannes Kepler derived three mathematical laws for planetary motion in the early 1600's. This work was based on measurements made before the advent of the telescope using instruments called quadrants and sextants. Oh, and before the advent of the computer as well.



Kepler's third law of Planetary Motion captures the relationship between the distance of planets from the Sun and their orbital periods. It states that the square of the orbital period of a planet is proportional to the cube of the semi-major axis of its orbit. If we assume (for convenience) that the period P is measured in Earth-years and the orbital radius R is measured in Astronomical Units (AU), then the equation is simply this:

Mathematically:

$$P^2 = R^3$$

where P is the period measured in Earth-years, and R is the semi-major axis (orbital radius) in Astronomical Units (AU).

Write a program to calculate the orbital radius of a planet given its orbital period. To solve this problem you may need to know that you can express square roots and cube roots as fractional exponents. For example:

$$x^{1/2} = \sqrt{x}$$

$$x^{1/3} = \sqrt[3]{x}$$

Input

Each line of input is the orbital period (P), in years, of a body orbiting the sun. The input ends with a value of zero.

```
1.8808
4.60
0.615198
30.07
0
```

Output

For each period, the program must print the semi-major axis (R) in AU. The answer must be accurate to within +/- 0.01 AU.

```
1.523679
2.7668
0.723327
9.6699
```

Problem 9 A Sorted Affair [6 points]

Summary

Q: How do you win HP CodeWars?
A: Everyone's a winner at HP CodeWars.

Seriously, if you want to win you have to score the most points, right? How do you think the judges keep track of the points? One simple way would be to record the team number and point value every time a program is judged to be correct. Then at the end of the contest a program could calculate each team's total and print the top five teams. Doesn't that sound easy?

Write a program to determine the top five teams using raw score data.

Input

Each line of input represents a single correct program using two integer values: a three-digit team number and a point value between one and twenty-one. The end of input is signaled by two zeroes.

```
123 1
354 2
213 1
354 5
112 2
508 1
481 5
508 5
481 3
354 7
508 9
112 3
```

```
(continued...)
481 1
987 3
246 1
213 2
354 6
742 1
354 13
987 15
213 8
987 7
833 1
```

```
354 4
742 2
508 3
742 3
642 3
213 5
508 2
354 3
642 1
213 11
0 0
```

Output

The program must print the top five teams, in order. Each line of output must have the team rank, the team number, and the total number of points scored by that team. Lucky for you there will be no ties.

```
1 354 40
2 213 27
3 987 25
4 508 20
5 481 9
```

Problem 10 EBCDIC

[7 points]

Summary

Extended Binary Coded Decimal Interchange Code (EBCDIC) is an 8-bit character encoding used mainly on IBM operating systems. EBCDIC was devised in 1963 and 1964 by IBM and was descended from the 6-bit code used with punched cards of the late 1950's and early 1960's.

EBCDIC Hexadecimal values

```
  a=81 b=82 c=83 d=84 e=85 f=86 g=87 h=88 i=89 j=91 k=92 l=93 m=94
  n=95 o=96 p=97 q=98 r=99 s=A2 t=A3 u=A4 v=A5 w=A6 x=A7 y=A8 z=A9
  A=C1 B=C2 C=C3 D=C4 E=C5 F=C6 G=C7 H=C8 I=C9 J=D1 K=D2 L=D3 M=D4
  N=D5 O=D6 P=D7 Q=D8 R=D9 S=E2 T=E3 U=E4 V=E5 W=E6 X=E7 Y=E8 Z=E9
  space=40  period=4B  comma=6B exclamation=5A
```

A different character-encoding scheme was also first published in 1963, called the American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that was adopted for a wide range of computer systems and became the most commonly used character encoding on the internet. It was surpassed in 2007 by UTF-8, which uses ASCII as a subset. Your laptop, desktop, tablet, and phone almost certainly use ASCII encoding.

> DUDE: I thought they were just letters and stuff.
> TECHIE: Dude, each letter is represented by a number. It's called a code.
> DUDE: (nodding slowly) oh.....

But sometimes old technology lingers and to this day there are systems that store and retrieve data using EBCDIC. Modern mainframes (such as IBM zSeries) include processor instructions, at the hardware level, to accelerate translation between character sets.

Write a program to convert text from EBCDIC to ASCII.

Input

The first line of input will indicate the number of lines of EBCDIC. Each line begins with a decimal number of EBCDIC codes, followed by that number of EBCDIC codes.

```
3
12 C8 D7 40 C3 96 84 85 E6 81 99 A2 5A
17 E5 85 95 89 6B 40 A5 89 84 89 6B 40 A5 89 83 89 4B
21 E6 85 40 81 99 85 40 A3 88 85 40 83 88 81 94 97 89 96 95 A2 4B
```

Output

The program must print the text represented by the EBCDIC input. Each line of input should correspond to a line of output.

```
HP CodeWars!
Veni, vidi, vici.
We are the champions.
```

Problem 2 Running Out of Water [2 points]

Summary

A small town in the California desert, Dry Gulch, has an underground water storage tank that contains 10,000 gallons of water. Thankfully (given that it's another drought season), the tank has not been discovered by any nearby golf courses, water parks, or mineral water companies, and is still dedicated for use only by the residents of Dry Gulch.

Given a weekly usage rate for the town residents, calculate the number of weeks the water will last.

Input

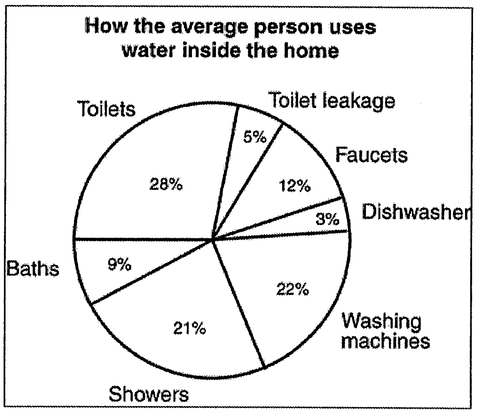
The input will consist of different weekly water usage rates ending with a zero.

```
1750
1000
4325
0
```

Output

For each line of input calculate the total number of full weeks that the water supply will last and print the output as follows.

```
1750 gallons per week will last 5 weeks
1000 gallons per week will last 10 weeks
4325 gallons per week will last 2 weeks
```



Problem 4 Scientific Notation

[3 points]

Summary

Scientists and engineers often deal with very large and very small things such as redwood trees and atomic particles. Our data values can similarly be very large or very small. In order to make it easier to read and write such numbers we use scientific notation.

Here's an example of a number written in scientific notation:

```
3.926 x 10^4 (which is 39260)
```

Write a program that will read in a scientific number as a base number and a power of 10 and calculate the equivalent decimal value.

Input

Each line contains a pair of numbers, B and E, where B is the base number and E is the power of 10. B is greater than 1 but less than 10, and E is in the range of -10 to 10. The last line of input is two zeros.

```
4.296 3
3.8 -2
1.8 2
2.8678 1
0 0
```

Output

For each line of input calculate the actual value rounding to 2 decimal places. Trailing zeros to the right of the decimal point are required.

```
4296.00
0.04
180.00
28.68
```