UTD Programming Contest for High School Students
April 14th 2012

- **Time Allowed: three hours.**
- **Each team must use only one computer - one of UTD's in the main lab.**
- **Answer the questions in any order.**
- **Use only Java, C, C++ or C#.**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from the named file specified in the problem header, and must output to System.out (cout for C/C++ programmers.)**
- **Do not access the web except to access Java or C/C++/C# documentation.**
- **Do not use any recording device containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client.**

Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

A. It's Tax Time!

Input File: A.txt
Runtime allowed = 3 seconds

In the country of Marginalado the tax code is very simple. If your gross income in any calendar year is below 20,000 Bargs, no taxes need be paid. If your gross income is more than 20,000 Bargs, then you must pay income tax given by the marginal tax rate r times the amount of your gross salary that is in excess of 20,000 Bargs.

Write a program to calculate the amount of tax due, given the gross income and the tax rate.

Input
The input will be made up of a sequence of datasets, each one on a single line containing two reals separated by a space. The first value gives the amount of gross income for a particular citizen and the second value gives the tax rate r ($0 < r < 1.0$). This list of datasets is terminated by End Of File.

The gross income will never be more than $1,000,000.00$ Bargs and will always be given with two figures after the decimal point.

Output
For each dataset, print the amount of tax due as a decimal value accurate to two places after the decimal point. Always print the two fractional digits, as shown below:

```
Sample Input      Sample Output
-----------------|--------------
13500.47 0.22    |0.00
100000.00 0.15   |12000.00
123456.78 0.125  |12932.10
```

B. Missing Card

Input File: B.txt
Runtime allowed = 3 seconds

One playing card is missing from a standard deck of 52 cards, but which one?

Input
The input file will contain multiple datasets, each one representing a sample deck of cards with one card missing. Each dataset comprises 51 lines containing descriptions of the 51 playing cards present in the deck. The cards will be listed in arbitrary order. Following each dataset will be a blank line.

Each card will be described by its value followed by the word "of" followed by its suit. For example, "Three of Hearts", "Jack of Diamonds", "King of Spades".

The input file will be terminated by End Of File.

Note that in the Sample Input below there are three datasets. Only a few lines of each dataset are shown.

Output:
For each dataset, output a line as shown in the Sample Output below.

```
Abbreviated Sample Input:     |Sample Output:
------------------------------|---------------------------
Three of Hearts               |The missing card is: Jack of Clubs
Two of Spades                 |The missing card is: Seven of Hearts
Ace of Diamonds               |The missing card is: Nine of Diamonds
Queen of Clubs                |
 . . .                        |
Ace of Hearts                 |
                              |
Three of Spades               |
Two of Diamonds               |
Four of Hearts                |
King of Clubs                 |
 . . .                        |
Two of Hearts                 |
                              |
Two of Hearts                 |
Ace of Spades                 |
King of Spades                |
 . . .                        |
Ten of Diamonds               |
```

C. Car Buyer Beware

Input File: C.txt
Runtime allowed = 3 seconds

When you buy a car and finance the purchase, the dealership figures out your monthly payments based on the price of the car, the down payment, and the interest rate. For this problem we are going to assume a simple interest system.

As an example, assume that I buy a car priced at $20,000 and I give the dealership $1,000 as a down payment. I owe $19,000 dollars.

If the interest rate is 5% per annum and I want to spread the payments over 36 months, then the total interest payments will be $19,000 times 3 years times 5% which equals $2,850, making a total loan amount of $21,850.

My monthly payments will be $21,850/36 = $606.94444'. This amount is always rounded up to the nearest cent: $606.95 per month.

Month by month the remaining loan amount is the total loan amount minus the sum of the payments I have made.

Unfortunately, as I drive the car off the lot, its value is $20,000 and I owe $21,850. If I total the car, the insurance company will only give me the value of the car after taking into account depreciation.

If the depreciation rate is a constant of 1% per month, then the car's value at some date is 1% less than it was the previous month. Its value month by month is $20,000$,19,800, $19,602$,19,405.98, etc. At some point the the amount that I owe will be lower than the value of the car. Until then, we say that, I'm "upside down" on this loan.

How many months will it be before I'm no longer upside down? Here is a table showing how the amount owed and the depreciated car's value change over the first 5 months. After 5 months I'm no longer upside down.

Month	Amount Owed	Car's Value	Difference
0	21850	20000	-1850
1	21243.05	19800	-1443.05
2	20636.1	19602	-1034.1
3	20029.15	19405.98	-623.17
4	19422.2	19211.9202	-210.2798
5	18815.25	19019.801	204.551

Input:
The input file contains information for several loans. Each loan description consists of one line of text containing the cost of the car, the down payment, the number of months of the loan, the percentage interest rate per annum on the loan, and the percentage depreciation rate per month on the car's value. The duration is an integer, the rest are

real numbers. All values are nonnegative, with loans being at most 100 months long and car values at most $75,000. The input will be terminated by End of File.

The first example below is the same as the one described above.

Output:
For each loan output the number of complete months before the borrower owes less than the car is worth.

```
Sample Input                          |Sample Output
--------------------------------------|-------------
20000.00 1000.00 36 5.0 1.0           |5
50000.00 1000.00 48 2.0 2.0           |14
70000.00 2000.00 48 5.0 2.0           |22
```

D. Automatic Editing

Input File: D.txt
Runtime allowed = 3 seconds

Some text-processing tools allow you to perform a sequence of editing operations based on a script. In this problem your program will read a single line of text and perform a series of string replacements on it, based on a fixed set of rules. Each rule specifies the string to find and the string to replace it with, as shown below.

```
Rule    Find    Replacement-String
---------------------------------
  1.    ban     bab
  2.    baba    be
  3.    ana     any
  4.    ba b    hind the g
```

To perform the edits for a given line of text, start with the first rule. Replace the first occurrence of the find string within the text by the replacement-string, then try to perform the same replacement again on the new text.

Continue using the same rule until the find string no longer occurs within the text, and then move on to the next rule. Continue until all the rules have been considered. Note that

(1) when searching for a find string, you always start searching at the beginning of the text,

(2) once you have finished using a rule (because the find string no longer occurs) you never use that rule again, and

(3) the case of the text is significant.

For example, suppose we start with the line

```
banana boat
```

and apply the rules above. The sequence of transformations is shown below.

```
Before        After
------------------------------
banana boat   babana boat
babana boat   bababa boat
bababa boat   beba boat
beba boat     behind the goat
```

Note that rule 1 was used twice, then rule 2 was used once, then rule 3 was used zero times, and then rule 4 was used once.

Input:
The input contains one or more test cases, followed by a line containing a zero that signals the end of the file. Each test case begins with a line containing the number of rules, which will be between 1 and 10. Each rule is specified by a pair of lines, where the first line is the find string and the second line is the replacement string. Following all the rules is a line containing the text to edit.

The find and replacement strings will each be at most 80 characters long. Find strings will contain at least one character, but replacement strings may be empty (indicated in the input file by an empty line). During the edit process the text may grow as large as 255 characters, but the final output text will be less than 80 characters long.

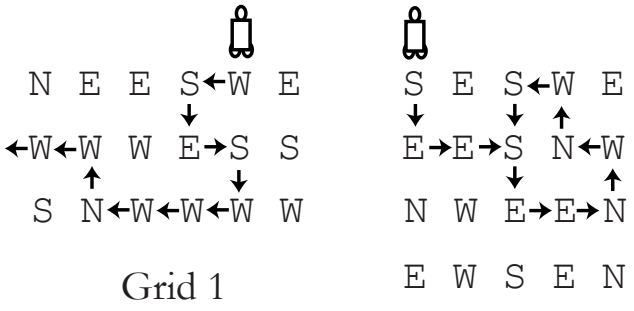
Output:
For each test case, output a line containing the final edited text.

The first test case in the sample input below corresponds to the example shown above.

Example Input	Example Output
4	behind the goat
ban	shoe or shop
bab	
baba	
be	
ana	
any	
ba b	
hind the g	
banana boat	
1	
t	
sh	
toe or top	
0	

E. Robot Motion

Input File: E.txt
Runtime allowed = 3 seconds



Grid 1

Grid 2

A robot has been programmed to follow instructions that are written on the floor in the squares of a grid. Each square specifies the direction of the step that the robot must take next. The possible instructions are:

N north (up the page), S south (down the page), E east (to the right on the page), W west (to the left on the page)

For example, suppose the robot is above column 5 of Grid 1 and starts south (downwards). The path followed by the robot is shown. The robot obeys ten instructions within the grid before leaving it.

Compare what happens in Grid 2: the robot obeys three instructions and then starts a loop of 8 instructions never leaving the grid.

You are to write a program that determines how many steps the robot takes to get out of the grid or how the robot loops around.

Input:
There will be one or more grids for the robot to navigate. The data for each is in the following form. On the first line are three integers separated by space characters: the number of rows in the grid, the number of columns in the grid, and the column number where the robot enters from the north. The columns are numbered starting at one at the left.

Then come the rows of the grid giving direction instructions. Each grid will have at least one and at most 10 rows and columns. The lines of instructions contain only the characters N, S, E, or W with no spaces. The end of input is indicated by a row containing 0 0 0.

Output:
For each grid in the input print one line of output. Either the robot follows a certain number of instructions and exits the grid on any one of the four sides or else the robot follows the instructions at a certain number of locations once, and then the instructions at some number of locations repeatedly. The sample input below corresponds to the two grids above and illustrates the two forms of output. The word "step" is always immediately followed by "(s)" whether or not the number before it is 1.

```
    Example Input  |    Example Output
-------------------|-----------------------------------
3 6 5              |10 step(s) to exit
NEESWE             |3 step(s) before a loop of 8 step(s)
WWWESS             |
SNWWWW             |
4 5 1              |
SESWE              |
EESNW              |
NWEEN              |
EWSEN              |
0 0 0              |
```

F. Cannonballs

Input File: F.txt
Runtime allowed = 3 seconds

It is common to see stacks of cannonballs in castles and fortresses all over the world. In this question cannonballs are stacked in pyramids. The base of a stack of cannonballs is made up of an equilateral triangle full of cannonballs, similar to the arrangement of pool balls at the start of a game. Layers are built on this foundation such that each cannonball in a certain layer touches three cannonballs in the layer beneath it. Each layer has as many cannonballs as possible that satisfy this touching rule. His Majesty's Navy has a huge warehouse in which it intends to store many stacks of cannonballs. Given the distance from the floor to the ceiling of the warehouse, how many cannonballs can there be in each stack? Assume that cannonballs are 1ft in diameter.

To help you answer this question it might be useful to know the formulae for the height of a tetrahedron in terms of the length, a, of one of its edges:

$$H = \frac{\sqrt{6}}{3}a$$

It may also be useful to know that the number of cannonballs in a stack of L levels is given by:

$$N = \frac{L(L+1)(L+2)}{6}$$

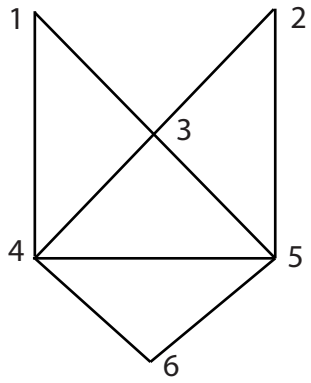
Input
The input will begin with an integer D on a line by itself giving the number of datasets to follow. Each dataset will comprise one line of text containing the height H of the warehouse, ($10.00 \leq H \leq 1000.00$) ft.

Output
For each dataset output one line of text of form, "The maximum number of cannonballs in a stack is M" where M is the (integer) number of cannonballs.

```
Sample Input              |Sample Output
--------------------------|------------------------
3                         |The maximum number of cannonballs in a stack is 364
10.00                     |The maximum number of cannonballs in a stack is 4960
25.00                     |The maximum number of cannonballs in a stack is 39711
50.00                     |
```

G. Don't Lift Your Pencil

Input File: G.txt
Runtime allowed = 3 seconds



Can you draw the figure above without lifting your pencil from the paper and without retracing any line? What if you have to start and end at the same point? If so, the path that your pencil follows is called an "Euler Circuit". Euler's Theorem states that there is an Euler Circuit if and only if every vertex has an even degree (number of lines joining it). For example, vertices 3, 4, and 5 above have degree 4, while vertices 1, 2 and 6 have degree 2.

Input:
The first line of the input contains a single integer N giving the number of problems to follow. Each problem description begins with an integer M on a line by itself giving the number of connecting lines in the figure. Then M lines of text follow, each one describing a single connecting line. Each of these lines contains two integers separated by a space. The integers represent the vertex numbers, 1, 2, 3, . . . There will be less than 100 connecting lines in each problem. You can assume that there is path between every pair of vertices.

Output:
If there is no Euler Circuit, print the string "No Euler Circuit" on a single line.

If an Euler Circuit exists, print the string "Euler Circuit Exists" on a single line.

Sample Input	Sample Output
3	No Euler Circuit
5	Euler Circuit Exists
1 2	Euler Circuit Exists
1 3	
1 4	
2 4	
3 4	
12	
1 2	
1 3	
2 3	
2 4	
2 5	
3 4	
3 6	
4 5	
4 6	
5 7	
5 6	
6 7	
3	
1 2	
2 3	
1 3	