PLU February 2014 Programming Contest

Advanced Problems

I. General Notes

1. Do the problems in any order you like.

2. Problems will have either no input or will read input from standard input (stdin, cin, System.in -- the keyboard). All output should be to standard output (the monitor).

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

Number	**Name**
Problem 1	Coupon1
Problem 2	AcidBase
Problem 3	Snake2
Problem 4	Coupon2
Problem 5	Focus
Problem 6	Rummikub
Problem 7	Golden
Problem 8	Quadrilateral
Problem 9	Tongues
Problem 10	Acronyms
Problem 11	Clean Up
Problem 12	King Me

Good luck!

1. Coupon1

Problem Description: You have some coupons that are 20% off any item. Given a price of an item, calculate the new price.

Input: The first line consists of the number of data sets. Each subsequent line contains one price.

Output: Show each price with a dollar sign to the nearest cent.

Sample Input:

```
3
100.00
59.99
20.00
```

Sample Output:

```
$80.00
$47.99
$16.00
```

2. AcidBase

Problem Description: There is no longer a STAAR test in chemistry—the pressure is off! You can just learn chemistry for the sheer fun of learning about the natural world (and to prepare for college)! Celebrating this fact, let's do a program on acids, bases, and pH.

The pH of a solution is a measure of the ionization of water to form H^+ (acids) or OH^- (bases).

There are four important variables (pH, pOH, H^+, and OH^-) and two important equations:

$$pH = -\log(H^+)$$

$$pOH = -\log(OH^-)$$

Here the logs are logs base 10. The variables pH and pOH are also related by the equation:

$$pH + pOH = 14$$

Input: The first line consists of the number of data sets. Each data set has either the H (representing H^+) or OH (representing OH^-) concentration.

Output: Print out the pH rounded to the nearest hundredth.

Sample Input:

```
3
H = 0.001
OH = 0.001
H = 0.012
```

Sample Output:

```
3.00
11.00
1.92
```

3. Snake2

Problem Description: Many people have played the snake game (a few years ago Nokia phones came with snake on it, sometimes when watching YouTube videos you could pause the video and press the left arrow to start the game…). Let's simulate that game!

For this program, you will be given a matrix (15x15) of spaces and letters. The snake will initially consist of 3 contiguous 'X' characters, and the food pellets are represented by the 'F' character. The object of the game is to eat food pellets and grow, without hitting the boundary (stay within the size of the matrix).

For this program, you need to keep track of how the snake grows, how many pellets it eats, and whether the game ends or not. The game will end if either the snake goes outside the playing area OR it runs into itself.

The input will be a string of 20 of the characters "UDLRO" standing for Up, Down, Left, Right, and Onward…continue in the same direction. For each input, determine either how many food pellets are eaten, or if it is game over (by leaving the matrix). The game will restart with the same board configuration of snake and pellets. The snake will start to move to the RIGHT at the beginning of the game, and the snake will be horizontal.

Here is an example of the snake moving with the following input string:

```
UROOOUOOLOOOOOUOOOOO
12345678901234567890
```

The first 5 moves made are in bold below (**12345**). On the 8th move, the snake would eat the pellet! Then, you would turn left and continue to eat the next pellet straight ahead (move 14).

```
 0                    X
 9                    X
 8                    X
 7                    X
 6           --->     X
 5F32109F             54321098
        7                    7
        6                    6
    12345                12345
  XXX    F                   F
```

Due to eating two pellets, the snake would be 5 X's long and be continuing upwards

(Continued on next page…)

(Problem 3 contin.)

Input: The first 15 lines consists of the matrix (15x15 characters). The next line will consist of the number of data sets. Each data set will consist of a line of 20 characters (UDLRO).

Output: For each data set, print out "N pellets" or "GAME OVER". Then print out the final game board. Separate games with a blank line.

Sample Input:

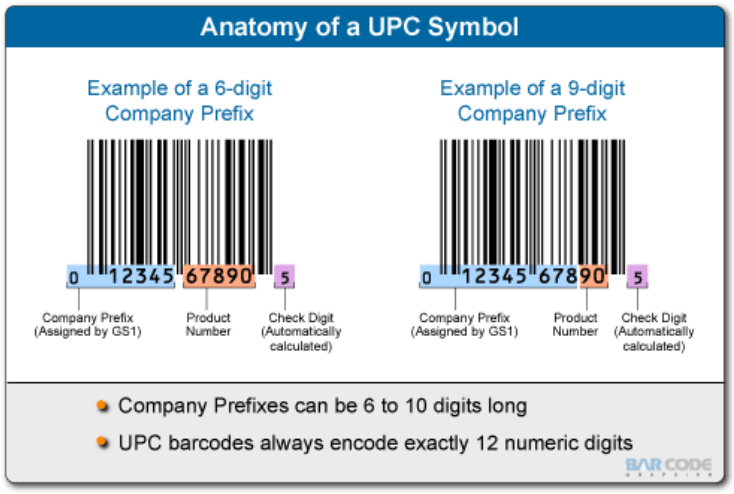
```
F               F
         F     F
 F
 F

  F      F


  XXX     F


       F  F

               F
2
UROOOUOOLOOOOOOUOOOO
UOOOOOOOOOOOOOOOOOOO
```

Sample Output:

```
4 pellets
FX              F
 X       F     F
 X
 X
 X
 XX


          F


       F  F

               F


GAME OVER
F   X           F
    X   F     F
 F
 F

  F      F


          F


       F  F

               F
```

4. Coupon2

Problem Description: When you go shopping and have coupons, you can only get discounts on the items you actually purchased. Before UPC codes and scanners, it was up to the cashier to check for purchases and to verify if the coupons were for valid purchases. Here is some information on UPC codes:



(http://cdn.vouchercodes.ca/wp-content/uploads/2012/05/anatomy-of-a-upc.jpg)

In this program, you will have a receipt showing the UPC codes and prices. Then you will be given a list of coupons with UPC code and discount. Apply the discount to valid purchases. The UPC codes will be a 12 digit numeric sequence and the price will have no dollar sign on input.

Input: The first line consists of the number of data sets. The first line in each data set will consist of X and Y, the number of purchases and number of coupons. The first X lines of each data set consist of the UPC code and the price. The next Y lines consist of the UPC code and the decimal of the discount (0.20 = 20% off). X and Y will be at least 1 and at most 20, with Y <=X.

Output: For each data set, print out the total of the purchases. If there are invalid coupons on the purchase, print out "INVALID COUPONS" on the next line followed by the UPC codes of the invalid coupons, each on one line.

(Continued on next page…)

(Problem 4 contin.)

Sample Input:

```
3
1 1
000000123456 19.99
000000123456 0.10
5 2
000000123456 19.99
000000123457 10.00
000000123458 20.00
000000123459 30.00
000000123460 40.00
000000123456 0.10
000000123459 0.20
5 2
000000123456 19.99
000000123457 10.00
000000123458 20.00
000000123459 30.00
000000123460 40.00
000000123456 0.10
000000999999 0.20
```

Sample Output:

```
17.99
111.99
117.99
INVALID COUPONS
000000999999
```

5. Focus

Problem Description: There is no longer a STAAR test in physics—the pressure is off! You can just learn physics for the sheer fun of learning about the natural world (and to prepare for college)! To celebrate this fact, here's a program on lenses.

Lenses use refraction of light to create magnified images. The distance to the object (p) and the distance to the image (q) are dependent on how "curved" the lens is, called the focal point (f). Using the following relationship:

$$\frac{1}{f} = \frac{1}{p} + \frac{1}{q}$$

For this program, given the object distance (p) and image distance (q), find the distance to the focus (f).

If you are interested in this equation, look up the Khan Academy (http://www.khanacademy.org) on this topic. He has a cool derivation of this formula using similar triangles.

Input: The first line consists of the number of data sets. Each data set is on one line and contains two integers (p and q).

Output: For each data set, print out "f = ___" with the distance rounded to the nearest tenth.

Sample Input:

```
3
5 6
5 5
10 15
```

Sample Output:

```
f = 2.7
f = 2.5
f = 6.0
```

6. Rummikub

Problem Description: In the game of Rummikub® you get tiles numbered from 1-13 of four different colors (red, blue, orange, and black). The purpose of the game is to make groups or runs from your 14 tiles. Groups are 3 or 4 of a kind of different color tile, and runs are three or more consecutive tiles of the same color. The point value of a group or run is the sum of point values of the tiles. The object is to play all of your tiles. This is like the card game gin or gin rummy. What a great program to write!

The input will have a color character (A,B, C and D) followed by a 1-2 digit number (1-13). So, a group would look like A5, B5, C5; a run would look like A5, A6, A7.

For this program, consider only the following objectives:

- determine whether or not your hand can make groups or runs
- determine the maximum point value of each combination

For example, the following 14 tiles:

A5 B5 C5 A6 A7 B13 A12 C1 C9 D1 D2 D3 D7 D12

can have the following groups: A5 B5 C5 (15 points)
or the following runs: A5 A6 A7 (18 points) and D1 D2 D3 (6 points)
So the maximum point value for a group or run is 18 points.

Input: The first line consists of the number of data sets. Each data set is on one line and contains 14 strings (a character followed by a 1 or 2 digit integer). Assume the tiles will be in random order.

Output: For each data set, print out either "NO GROUPS OR RUNS" or a sorted list of the group or run with the highest point score. This sorted list should have spaces and have the points as the last item on the output line.

Sample Input:

```
3
A5 B5 C5 A6 A7 B13 A12 C1 C9 D1 D2 D3 D7 D12
A1 A2 B1 B2 C5 C7 C9 C11 B6 B8 B10 B12 D3 D4
D13 C9 C13 A1 A3 A4 A5 A9 A13 B1 B2 B11 B12 B13
```

Sample Output:

```
A5 A6 A7 18
NO GROUPS OR RUNS
A13 B13 C13 D13 52
```

7. Golden

Problem Description: There are many special irrational numbers in nature. One is the "golden ratio" represented by the Greek letter phi, φ, which has a value of

$$\varphi = \frac{1 + \sqrt{5}}{2} = 1.61803399 \ldots\ldots$$

Given 2 real numbers, determine whether their ratio is equal (or close to) the golden ratio. For this program, use ±1% as the tolerance.

Input: The first line will contain the number of data sets. Each data set consists of 2 real numbers (with or without decimals).

Output: Either print out "golden" or "not" for each data set.

Sample Input:

```
3
1.61803399 1
1.699 1.0
1 1
```

Sample Output:

```
golden
not
not
```

8. Quadrilateral

Problem Description: A quadrilateral is a 4-sided figure. For this program, print out a 4 sided figure using the capital letter X. The largest side will be 20, and the smallest side will be 1.

Input: The first line consists of the number of data sets. Each data set consists of 2 integers, the length and width of the quadrilateral.

Output: Print out the quadrilateral using the capital letter X, each quadrilateral separated by a blank line.

Sample Input:

```
3
2 3
5 1
20 8
```

Sample Output:

```
XX
XX
XX

XXXXX

XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
```

9. Tongues

Problem Description: Gandalf's writings have long been available for study, but no one has yet figured out what language they are written in. Recently, due to programming work by a hacker known only by the code name ROT13, it has been discovered that Gandalf used nothing but a simple letter substitution scheme, and further, that it is its own inverse, the same operation scrambles the message as unscrambles it.

This operation is performed by replacing vowels in the sequence

(a i y e o u)

with the vowel three advanced, cyclically, while preserving case (i.e., lower or upper). Similarly, consonants are replaced from the sequence

(b k x z n h d c w g p v j q t s r l m f)

by advancing ten letters. So for instance the phrase

One ring to rule them all.

translates to

Ita dotf ni dyca nsaw ecc.

The fascinating thing about this transformation is that the resulting language yields pronounceable words.
For this problem, you will write code to translate Gandalf's manuscripts into plain text.

Input: The input will contain multiple problem sets. Each problem set consists of a single line containing up to 100 characters, representing some text written by Gandalf. All characters will be plain ASCII, in the range space (32) to tilde (126), plus a newline terminating each line. The end of the input is denoted by the string "EOI". The string "EOI" will be on a single line by itself and is not a problem set.

Output: For each problem set, print its translation into plaintext. The output for each problem set should contain exactly the same number characters as the input.

Sample Input:

```
Ita dotf ni dyca nsaw ecc.
EOI
```

Sample Output:

```
One ring to rule them all.
```

10. Acronyms

Problem Description: There are many acronyms that have the same letters with different meanings. For example, ISS could mean International Space Station, In School Suspension, Interstate Sporadic Squabbles, but it could not be Institutional Saliva Tester (IST, not ISS) nor could it be Intermural Sports Situational Comedies (ISSC not ISS).

Determine whether or not phrases fit the acronym, with the correct number of words and beginning letters. All words will begin with capital letters.

Input: The first line contains the number of data sets (maximum of 10). Each data set contains the acronym and the number of possibilities (maximum of 10) on one line, followed by each possibility on a separate line.

Output: Show the acronym on one line followed by the phrases that match that acronym each on one line.

Sample Input:

```
4
ABC 4
American Broadcasting Company
Another Boring Computer Program
Alpha Beta Gamma
Another Brilliant Csteacher
UIL 3
Union Leaders Invitation
University Interscholastic League
Upstanding Important Leaders
CIA 4
Central Intelligence Agency
Computer Image Advertising
Coming In After Christmas
Clubbing Intelligent Aardvarks
TBBT 7
The Big Bang Theory
The Big Bama Tide
Two Boolean Byte TreeMaps
Thing Without An Important Name
Tuning Big Brass Tubas
This Bytes Big Boogies Totally
Disk Operating System
```

(Continued on next page…)

(Problem 10 contin.)

Sample Output:

```
ABC
American Broadcasting Company
Another Brilliant Csteacher
UIL
University Interscholastic League
Upstanding Important Leaders
CIA
Central Intelligence Agency
Computer Image Advertising
Clubbing Intelligent Aardvarks
TBBT
The Big Bang Theory
The Big Bama Tide
Two Boolean Byte TreeMaps
Tuning Big Brass Tubas
```

11. Clean Up

Problem Description: You work for the IT department of a major league baseball team. Baseball has an almost infinite number of stats—what a great job for a computer program!

You will write a program to determine who will bat 4th in the batting order. This is called the "clean-up" position and some managers use the person with the best batting average.

You will be given the roster (with their hits and at bats). The clean-up batter will be fourth and have the highest batting average.

You will also be given a series of hits and at bats which may change the best current batting average. Here are three possibilities:

1. The clean-up batter still has the greatest average (no change!)
2. One of the first 3 batters now has the greatest—switch positions with the clean-up hitter.
3. One of the last 5 batters now has the greatest—insert the newest batter at the fourth position and the 4th becomes the 5th, the 5th becomes the 6th… until the old batting position is reached.

Input: The first 9 lines consist of the current roster (last name only) and their past hits and at bats. The next 9 lines consist of the hits and at bats for the next series of games (in the same order as the roster).

Output: Show the new roster with the highest batting average in the clean-up position.

Sample Input:

```
Uecker 15 150
Ruth 50 150
Canseco 45 150
Beltre 55 150
Cruz 51 150
Kinsler 44 150
Andrus 40 150
Moreland 35 150
Darvish 10 150
0 10
5 10
0 10
0 10
9 10
5 10
5 10
5 10
1 10
```

(Continued on next page…)

(Problem 11 contin.)

Sample Output:

```
Uecker
Ruth
Canseco
Cruz
Beltre
Kinsler
Andrus
Moreland
Darvish
```

12. King Me

Problem Description: In addition to Rummikub and other board games, my daughters and I play checkers. First I am teaching them the rules. Then I teach them some strategy and thinking skills. I let them win more than I do to keep it fun, but I show them how they could do better in the next game.

In the game of checkers, you can jump over the opponent checkers diagonally if the next diagonal space is empty. Once you reach the other side of the board, you get "kinged" and the checker can now move backwards. You can also do multiple jumps going forward or backward. Let's write a program to determine which move you should make — which move would have the highest number of jumps!

You will be given a checkerboard in the middle of a game. You will be the red player (at the bottom of the board) playing against black (at the top). Find the position of the red Kinged checker (row and column in the matrix) that would have the maximum number of jumps. There will always be at least one jump. No two red kings will tie for equal maximum jumps, but there may be two identical jumps for the same king. Since there are 12 checkers, it is theoretically possible to have a maximum of 12 jumps. However, to get across to the other side of the board, you usually jump several checkers to get "kinged." Therefore you might get 3-4 checkers on a really good jump, or maybe 6 checkers on an exceptional move playing against a sub-par opponent!

For example, on the following checkerboard, the K at row 1 and column 3 can jump once down and right, but twice down and left.

```
B B
   K
B B B R
       R
  B
 R     R
R B R
 R R R R
```

B		B					
			K				
B		B		B		R	
							R
		B					
	R						R
R		B		R			
	R		R		R		R

Input: The first line contains the number of data set (checkerboards). Each checkerboard has 8 lines of 8 characters each (one of four letters, R, B, K or space)

Output: For each board, print out the position of the K checker with the maximum number of jumps.

(Continued on next page…)

(Problem 12 contin.)

Sample Input:

```
3
    K B
     B B
B   B
     B B
B B R
 R R R R
R R   R
   R R
B B
   K
B B B R
       R
  B
 R     R
R B R
 R R R R
K     R
 B R B B

 B B B

 B B B
R
       R
```

Sample Output:

```
0 4 1
1 3 2
0 0 6
```