Fall 2012 PERL Programming Competition

For in the Game of Nerds, you program, or you die. There is no middle ground.

1 Overview

Welcome to the Fall 2012 PERL Programming Competition! This is a contest wherein you compete amongst your fellow students, and professors, in an effort to solve a set of programming problems as fast as possible. Whoever solves the most problems by the end of the contest wins! In the event of a tie, the win goes to whoever submitted their final problem earliest.

2 Problems

There are seven problems that vary in difficulty. They are not arranged in order of difficulty. Each problem has a general description, details on the formats for input and output, and example input and output. Output of your program must match that of the correct output *exactly*. If there is even an extra space in your program's output, it will be rejected. All input is done from standard input and all output is done to standard output. *You should not open any files inside of your program.* You can solve the problems in C, C++, Java, Python, Ruby, or R. You must save your code using the standard extension for your language of choice (.c .cpp .java .py .rb .R) respectively.

3 Rules

You are allowed to have any printed material such as books or printouts. You are also allowed to access standard library documentation for your language of choice. *No other use of the internet is permitted.* You are also not allowed to copy and paste code from any source be it a thumb drive, or another file that you have. Only one computer is allowed to a team!

4 Logging In

To log in to the system, direct your browser to http://rosemary.umw.edu/mooshak/. You will then need to enter your team name and password which is provided to you. This will bring up the interface for you to submit solutions to the problems, ask questions (which will be broadcast to all participants), and see how your team is doing.

5 Submitting

To submit a solution, first make sure that you select the correct problem letter on the top of the site. Then click the "Choose File" button. Then choose the file that contains your source code (be sure it has the correct extension). After that it will be submitted for judging. At some point, it will receive a result indicating whether or not it was correct or not.

BRACE YOURSELF
THE PROGRAMMING COMPETITION IS COMING
BUT DO NOT TURN OVER THIS SHEET UNTIL THE CONTEST STARTS

A - Team Ordering

Anders Yronwood, Lord of Lemonwood, in the southern land of Dorne, has a headache. He has many children and just now, quite a few of them want to play the game “Come into My Castle.” This game consists of two teams (the rest of the rules are unimportant right now). The only thing is, the kids are all fighting to see who is on whose team and the noise is driving poor Lord Yronwood absolutely insane. He takes to the task of making the teams but neither his screaming, fighting kids, nor his gods forsaken headache can quell his methodical nature: he wants a precise way to split his kids up, a system that works every time, **no matter the number of kids, nor their ages**. As House Yronwood's Maester, it is up to you to create this systematic method of splitting his kids up into the two teams each time so they can play “Come into My Castle.”

Make a program that orders Lord Yronwood's kids in the following way: one team will have kids with odd-numbered ages, in ascending order (i.e., 1, 5, 7, 7, 9, 11, etc.) and the other team will have kids with even-numbered ages, in descending order (14, 12, 10, 10, 8, 6, etc).

Input:
The child's name with a colon, then an empty character (space) followed by the age, an integer.
Child's name: age. A period “.” will signify the end of the input

Sample Input:
```
Anders: 9
Gwyneth: 14
Mikel: 10
Yareen: 13
Valen: 8
Doran: 5
.
```

Output:
The first line will say Team 1 and will have the ascending, odd-number aged kids, then the next lines will have each child's name, a colon, a space, and then the integer age. Then an empty line and then after the empty line, it will say Team 2. The following lines will have the same for the descending, even-numbered age kids.

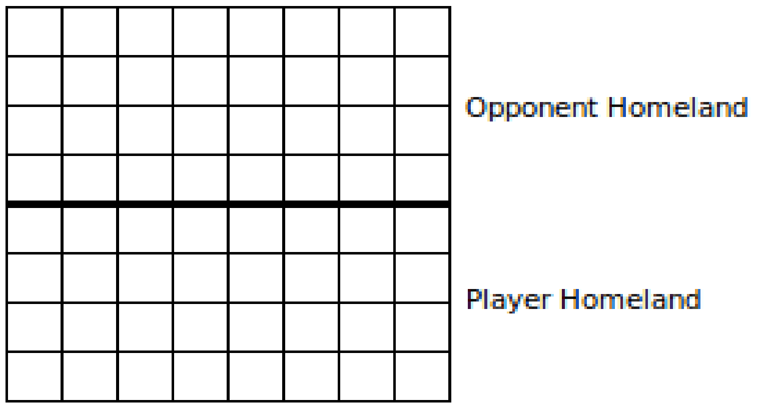
Sample Output:
```
Team 1
Doran:5
Anders: 9
Yareen: 13

Team 2
Gwyneth: 14
Mikel: 10
Valen: 8
```

B - Cyvasse

Cyvasse is a game that is widely played in the city of Volantis by lords, ladies, and commoners alike. And Tyrion Lannister is positively obsessed with it. The problem is, he is awful at it – for now. Never one to back down from an intellectual challenge (or a night of drunken debauchery, but that isn't important just now), he is trying to determine if a certain tile-sized area is accessible by an opponent. The rules are below. Help Tyrion out!

The game consists of a board 8 tiles high by 8 tiles wide. The board is divided into two halves, called "homelands" each four tiles high by 8 tiles wide.

At the start of the game, a screen is placed between the two homelands so the opponents cannot see each other's homeland. Each player arranges their tiles on their homeland as they like, at which point the screen is removed.



Cyvasse Board

The 32 tiles are:
- 14 grass tiles (G)
- 6 mountain tiles (M)
- 6 forest tiles (F)
- 5 water tiles (W)
- 1 fortress tile (X)

After the board is setup, each player moves their units (including rabble, spearmen, crossbowman and cavalry) across the board in an effort to capture the opponents fortress. The units can move in any cardinal direction, but not diagonally.

Mountain tiles are impassable to the player units in a game of Cyvasse. It is illegal to completely surround the fortress with mountain tiles as this would make it impossible for your opponent to win. It is also illegal have the fortress inaccessible due to a combination of mountain tiles and the edges of the board. Forest and water tiles slow down units, but are not impassable.

You must write a program that reads in an 8 tile by 4 tile homeland and decides whether or not Tyrion's fortress tile is accessible to the other player's pieces.

Input:
The first line of input is a positive number representing how many test-cases there will be. Each test-case consists of four lines each with eight characters representing the tiles of the homeland. The characters indicate the type of tile using the letters in parenthesis next to each type of tile above. Test-cases are separated by lines consisting of "-".

Output:
Output is to consist of one line for each test case, either "Fortress accessible." or "Fortress inaccessible.".

Sample Input:

```
2
GGFGGWFG
MMWWMGFG
XMWFFGMG
GMGGWGGF
-
GGMMFFGG
FGGWMMGG
MWGHXMWW
GGFFWGGG
```

Sample Output:

```
Fortress inaccessible.
Fortress accessible.
```

C - Coinage

The Seven Kingdoms use a system of currency based on metal coins. The coins in use are: the gold dragon, the silver stag, the copper star and the copper penny. 1 gold dragon is worth 210 silver stags. 1 silver stag is worth 7 copper stars. 1 copper star is worth 8 copper pennies.

Lord Petyr Baelish, called by some as Littlefinger (though never to his face), is Master of Coin on King Robert Baratheon's Small Council. He is known to make gold appear from thin air. Of course, gold never appears from thin air. Baelish actually has a vast network of brothels scattered across the largest cities in Westeros (especially King's Landing). The Crown has essentially been taking loans out from Baelish, with coin acquired through the various, seedy, flesh markets he owns and operates.

Because of the high demand, the brothels are very busy and you, Lord Baelish, would like to write a program to calculate the change on each purchase of flesh automatically. Because you are very particular person, you must make a program that always give as many high denomination coins as possible. For example, you should never give a “customer” more than 7 copper pennies, but exchange them for a higher valued coin instead.

Input:
The first line of input will be an integer, N, for the total number of “sessions” with a lady of ill repute. After that will be N lines which each represent one session. Each session line consists of the session price, followed by a space, followed by the amount of money that the customer paid. The monetary values are represented with four values separated by commas, one for each denomination from the most valuable gold dragon to the least valuable copper penny.

Output:
Output should consist of one line containing the change for each session line in the input. The change should be given in the same format as the session price and paid amount. If the customer did not provide enough money to enjoy a night out with one of Baelish's women, you should print "Go home to your wife" instead.

Example Input:
```
3
7,100,5,4 8,0,0,0
4,23,19,88 3,23,55,16
0,2,5,6 4,0,0,0
```

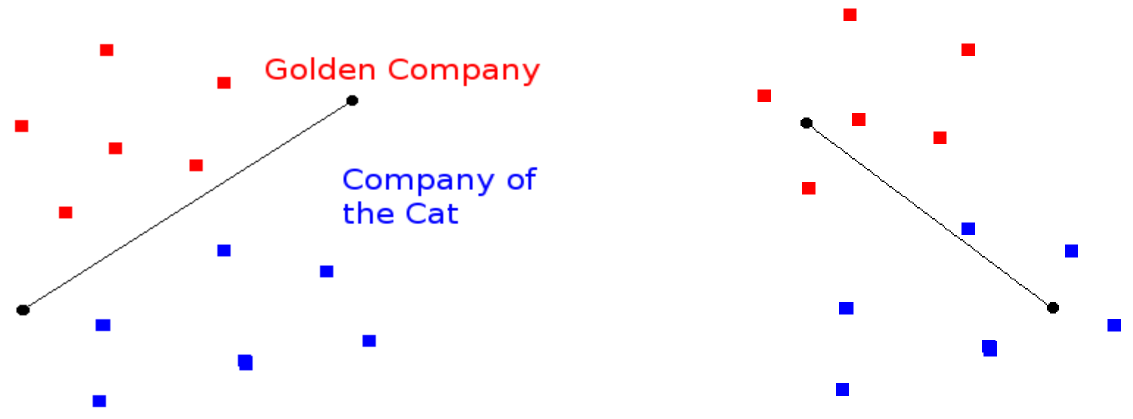
Example Output:
```
0,109,1,4
Go home to your wife
3,207,1,2
```

D - Camps

The Disputed Lands of Essos, across the Narrow Sea, have always been a hotbed for conflict. While there is little there of actual value, the Free Cities of Myr, Tyrosh, and Lys have always been embroiled in some conflict for these lands in some way, shape, or form. The Free Cities are structured differently from the fiefdoms and landed gentry-based armies of Westeros and as such, have never had Lords to call upon to fight their wars. Instead, they employ the fickle and ubiquitous mercenaries known as sellswords. These sellswords many times come together to form groups with bombastic-sounding names like the Brave Companions or Company of the Cat.

So, it's unsurprising that now Lys and Myr are both at war. Again. Lys has employed the fabled Golden Company, while Myr had to find a cheaper route and employed the Company of the Cat. Unfortunately for you, there are but a handful of mathematicians in Lys and Myr. In order to defend the cities, each side of the conflict wants to know whether or not a wall can be built to separate the two sides in the conflict.

Each side's home camp consists of a set of outposts established at a location which is represented by an x,y coordinate. The Golden Company and the Company of the Cat each have a camp with a number of outposts. You must write a program that, given the location of each of the outposts of each camp, and the location of a wall, determines whether or not the wall will separate the two camps or not. The example on the left shows the two camps separated by a wall; the example on the right shows two camps *not* separated by a wall.



Input:

Input will consist of a number of test cases. Each test case consists of three sections. The first section will start with "GC:" followed by a number of lines with x,y locations that give the location of each Golden Company outpost. The second section will start with "CC:" followed by a number of lines with x,y locations giving the location of each outpost for the Company of the Cat. The third section will start with "Wall:" followed by two lines with x,y locations giving the start and end locations of the wall. Input is terminated with a $ character.

Each camp will have no fewer than one, and no more than 100 outposts. You can assume that a wall is built as a straight line and can be extended infinitely in both directios.

Output:

Output consists of one line for each test case that either says "Wall separates camps." or "Wall does not separate camps."

Example Input:

```
GC:
7,7
```

```
2,14
13,16
23,5
6,27
CC:
15,26
24,16
28,27
2,40
Wall:
2,34
30,4
GC:
6,12
18,10
12,4
16,22
CC:
30,15
27,5
34,9
Wall:
32,4
22,28
$
```

Example Output:

```
Wall separates camps.
Wall does not separate camps.
```

E - Mottos

The Great Houses are the most powerful of the noble houses of the Seven Kingdoms. They exercise immense authority and power over their vassals and the areas that they control. They are typically answerable only to the King on the Iron Throne. Each member of a Great House uses the House name as their last name.

Each of the Great Houses has a motto which serves as a rallying cry for members of the House during battle. You are to write a program that reads in a series of house names along with their mottos, then a series of people's names. Your program is to then print out the motto of each person in the input.

Input:
The first line of input contains two integers: N and M. N refers to the number of houses and M refers to the number of people. After this line, there will be N lines, one for each house. Each of these lines is in the format HouseName:Motto. The mottos may have spaces in them. After the N lines for each house, there will be M lines that each refer to one person. N and M will both be less than 100, each name will be less than 50 characters long, and you may assume every person given does belong to one of the houses given and has exactly one first name and one last name.

Output:
For each of the M people in the input, your program should output one line consisting of their name, a colon and then the house motto of that person.

Sample Input:
```
4 6
Tully:Family, Duty, Honor
Stark:Winter is Coming
Martell:Unbowed, Unbent, Unbroken
Baratheon:Ours is the Fury
Doran Martell
Eddard Stark
Hoster Tully
Stannis Baratheon
Robb Stark
Trystane Martell
```

Sample Output:
```
Doran Martell:Unbowed, Unbent, Unbroken
Eddard Stark:Winter is Coming
Hoster Tully:Family, Duty, Honor
Stannis Baratheon:Ours is the Fury
Robb Stark:Winter is Coming
Trystane Martell:Unbowed, Unbent, Unbroken
```

F - Longships

During the War of the Five Kings, Stannis Baratheon once commanded a sizable fleet in his struggle to oust whom he thought was the great pretender sitting on the Iron Throne: Joffrey Baratheon. Within his main fleet, he also had a small corps of sleek longboats that Stannis had captured when he smashed Victarion Greyjoy's Iron Fleet during the Greyjoy Rebellion. What Stannis needed his trusted Hand, Ser Davos Seaworth, to do is to supply Stannis's fleet from the nearby coastal areas.

Stannis Barathen is a systematic, meticulous and brittle man, so he himself devised a plan to distribute barrels of supplies among the longships to be picked up at secret, predetermined sites along the coast known only to the trusted captains of the longships. The problem is that longships are designed to carry lightly-armored corsairs who can quickly embark and disembark, then melt away into the salty mist of the sea. They are not large galleons that can hold tonnes of cargo. Your job as His Grace's trusted Master of Spies is to help Stannis Baratheon with his army's supply problem. Stannis, after heavily modifying the longships, is able to fit 24 barrels per longship. Make a program can figure out which sips should pick up barrels at which supply site according to Stannis's method while maintaining the cargo load limit stated above.

There are a number of predefined pick up locations, numbered 1 to N. At each pick up location it will list the number of barrels getting picked up in the longships. Each longship will start at the next pick up location with few enough barrels to fit on the longship. Then it will continue down to the next site with few enough barrels to fit, and so on, until the longship has 24 barrels or there are no more supply sites. No supply site can be split up into two longships – all of the barrels must be picked up at once. Then the next long ship will continue this process, and so on, until all of the barrels have been picked up. Your program must compute how many longships are needed, and which locations each one picks up from.

Input:
The first line of input is the number of pick up sites, N. Subsequent lines each represent one pick up location. Each line starts with the location number, followed by the number of barrels to be picked up at that site. Each site will have at least one, and no more than 24, barrels to pick up.

Sample Input:
```
25
1    3
2    5
3    10
4    24
5    9
6    7
7    20
8    2
9    10
10   4
11   6
12   9
```

```
13   8
14   11
15   14
16   2
17   1
18   10
19   4
20   9
21   3
22   10
23   11
24   3
25   2
```

Output:
Output consists of one line for each longship needed. Each line consists of the word "Longship" followed by the longship number, then the words "stops at", followed by a list of each pick up site the longship stops at separated by spaces.

Sample Output:

```
Longship 1 stops at 1 2 3 8 10
Longship 2 stops at 4
Longship 3 stops at 5 6 11 16
Longship 4 stops at 7 17 21
Longship 5 stops at 9 12 19
Longship 6 stops at 13 14 24 25
Longship 7 stops at 15 18
Longship 8 stops at 20 22
Longship 9 stops at 23
```

G - Ciphers

The Battle of the Redgrass Field, during the time of the Blackfyre Rebellion, is going all wrong. Aegor Rivers, also known as Bittersteel, has been harrying the forces of the rightful heirs to the Iron Throne all day long. Bittersteel has taken his group of mercenaries, the Golden Company, around the right flank of Baelor and Maekar Targaryen. No matter what moves Baelor and Maekar Targaryen make, Bittersteel and his brother Daemon Blackfyre seem to be able to counter each move with a timely counterstroke that undoes each tactic.

Yet all is not lost. Brynden Rivers, called Bloodraven, and bastard half-brother to the combatants, has remained a loyalist to the Throne and has found a way to end the battle. In order to carry out the maneuver, he needs the main body of the loyalist forces to shift and mask the movements of him and his corps of archers, the feared Raven's Teeth. Yet he cannot risk leaving his current position. So, the cunning Bloodraven sends a cyphered message to his half-brother Maekar.

Baelor’s Cipher

In Baelor’s Cipher you are given a plain text message and a secret key k and you encrypt that message by rotating each letter k positions wrapping around from ‘Z’ to ‘A’ as needed. In other words, if p is some plaintext (i.e., an unencrypted message), pi is the ith character in p, and k is a secret key (i.e., a non-negative integer), then each letter, ci, in the ciphertext, c, is computed as:

```
ci = (pi + k) % 26
```

So, for example, suppose that the secret key, k, is 13. The relationship between the plain and cipher characters are:

```
plain:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
cipher:  NOPQRSTUVWXYZABCDEFGHIJKLM
```

If my message, p, is “Be sure to drink your Summerwine.” We can encrypt that p with that k in order to get the ciphertext, c, by rotating each of the letters in p by 13 places:

```
plain:  Be sure to drink your Summerwine!
cipher: Or fher gb qevax lbhe Fhzzrejvar!
```

Note that we preserve case and only alphabetic letters are rotated - non-alphabetic characters pass through unencrypted.

Brandon’s Cipher

Brandon’s cipher improves upon Caesar’s by encrypting messages using a sequence of keys (or, put another way, a keyword). In other words, if p is some plaintext and k is a keyword (i.e., an alphabetical string, whereby A and a represent 0, while Z and z represent 25), then each letter, ci, in the ciphertext, c, is computed as:

```
ci=(pi +kj)%26
```

Note this cipher's use of kj as opposed to just k. And recall that, if k is shorter than p, then the letters in k must be reused cyclically as many times as it takes to encrypt p.

Here is an example. Suppose our plain text is "Meet me at the Northern Border of Westeros!!" and our key is "Catelyn". Then our encrypted text would be "Oexx xc nv tai Ymevhxvy Zbtdxv zd Jgsmicmf!!"

```
plain:  Meet me at the Northern Border of Westeros!!
cipher: Oexx xc nv tai Ymevhxvy Zbtdxv zd Jgsmicmf!!
```

You are to write a program to help Maekar decrypt encrypted text using Brandon's Cipher that was sent to him by Bloodraven.

Input:

The format of the input is:
keyword
n, the number of lines of encrypted text
line 1
line 2
…
line n
keyword
n
line1
line2
…
line n

And so on. The end of input will be marked with a key of "END".

Output:
You are to output one line of decrypted text for each line of encrypted text. Again, all non-alphabetic characters pass through un-decrypted and case should be preserved. There should be one blank line between messages.

Example Input:
```
Catelyn
2
Oexx xc nv tai Ymevhxvy Zbtdxv zd Jgsmicmf!!
Vhx rteuv wtxnf jkle qpcg ws.
Shae
3
Loec klt jgytl sa deqirisr ts kle e ehn fwoeevld, xolnxq pn eds.
Loiw ohs xzl fmjzt xate lw daw vleqwk opv lnsmnh xg no.
Aa wek ahi fpnxz feej vf wmtmij, hnh loe wwcerlo oj Tyar'k sijw.
END
```

Example Output:

Meet me at the Northern Border of Westeros!!
The night watch will meet us.

They set forth at daybreak to see a man beheaded, twenty in all.
This was the first time he was deemed old enough to go.
It was the ninth year of summer, and the seventh of Bran's life.