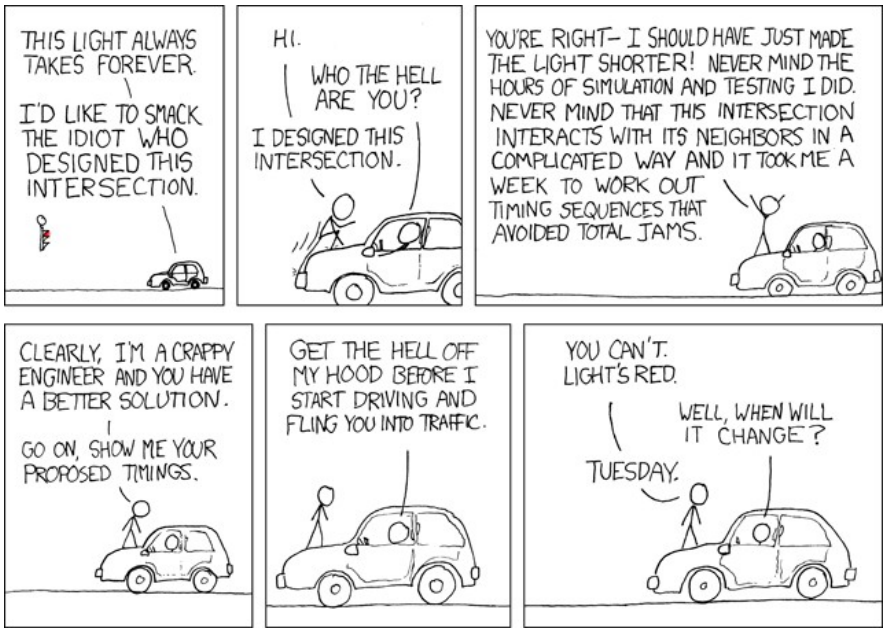
Problem Set (The XKCD Set)

Trinity University ACM
High School Programming Competition
April 10th, 2010

Problem 0 – Traffic Simulation



Introduction

It is common for civic planners to run simulations of traffic lights before installing the lights. These simulations use real or statistically generated arrival times for cars and consider different light timing settings to see which works best. One good metric for determining how well a potential timing works is to look at how long people have to wait at the light. For this problem, you will calculate average wait times for a light given the length of time between green cycles, how many cars can get through on a green, and information on arrival times of cars.

To keep this simple, make the following assumptions. Cars only get through if they arrive at or before the green cycle begins. At the beginning of the simulation, a green cycle has just ended and there are no cars. When the light turns green, all cars up to the number specified or the number in line get through.

Let's go through an example. Consider a light with a green cycle time of 100 seconds that lets 2 cars through each cycle and cars arriving at 30, 60, 90, 120, and 150 seconds. The first three cars arrive before the light turns green at their specified times. When it turns green at 100 seconds, the first two go through with wait times of 70 and 40 seconds respectively. The third car has to wait for the next cycle. During the next cycle the fourth and fifth cars arrive. At 200 seconds the light turns green again and the third and fourth cars get through with wait times of 110 and 80 seconds. When the light turns green again at 300 seconds, the fifth car gets through with a wait time of 150 seconds. The average wait time is therefore (70+40+110+80+150)/5=450/5=90 seconds.

Input

The first line will contain the number of sets (Integer, 1<= S<=50). Each set will have two lines. The first line gives the green cycle time (Integer, 1<=T<=1000), the number of cars that get through each green (Integer, 1<=N<=20), and the number of cars to consider (Integer, 1<=C<=100). The second line of each set will have C arrival times (Integer, 0<=A<=10,000). The arrival times will be in sorted

order. It is impossible for two cars to arrive at exactly the same time.

Output

For each set you output one line that says, "The average wait time for set # is #." The first # is the number of the set beginning with 1. The second # is the average wait time accurate to within one second.

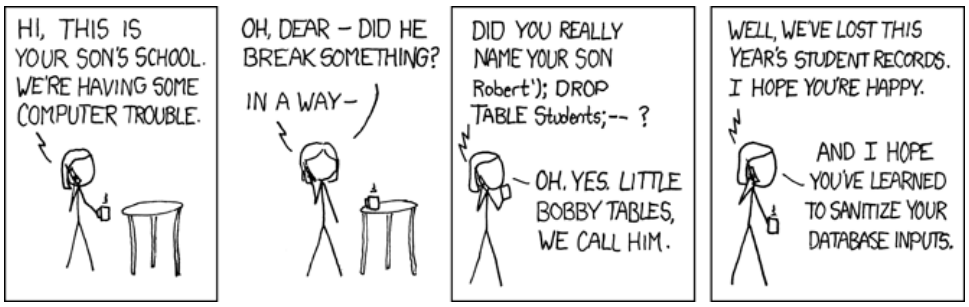
Sample Input

```
2
100 2 5
30 60 90 120 150
200 10 4
50 200 250 300
```

Sample Output

```
The average wait time for set 1 is 90.0
The average wait time for set 2 is 100.0
```

Problem 1 – Sanitizing Input



Introduction

You have been hired by the local Elementary school after some unwitting teacher entered a database command and deleted most of the student records. They've employed you to write a program that will be run on any database input that will ensure that any command entered will be free of characters that might cost them their data once again.

Input

The first line will contain the number of sets (1<=S<=50). On subsequent lines will be database inputs, which you will need to check, clean up, and then print back out. Words will be delimited by spaces. Data management commands are written completely in capital letters and need to be removed. Non-letter characters are also risky and should also be removed.

Output

One output line should be printed for each line of input. The output line should not have any special characters and all words that contain only capital letters or special characters should also be removed. There should be one space between each of the remaining words.

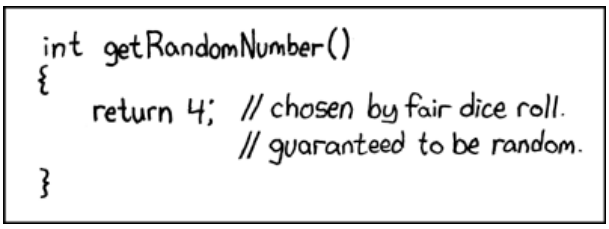
Sample Input

```
3
Robert');
add Robert FIND Smith ; DELETE TABLE STUDENTS;
rollProgram'); EXIT
```

Sample Output

```
Robert
add Robert Smith
rollProgram
```

Problem 2 – True Random Numbers



Introduction

Your boss at Reliable Programs Inc. would like you to write him a random number generator. However, your boss is using the random number generator to test several off the company's applications, so it needs to reliably print out the same set of numbers over and over. A common form of random number generators, called linear-congruential generators, work as follows:

$$X_{n+1}=(aX_n+b) \bmod c$$

Where X_n is the last random number generated (or seed, if one hasn't been), X_{n+1} is the next random number, and a, b, and, c are all numerical constants. In this case, your boss has asked you to use a = 17, b = 9, and c = 253.

Input

The first line will contain the number of inputs (Integer, 1<=I<=50). On subsequent lines will be two numbers, S and N (Integers, 0<=S=<10000, 1<=N<=25), which will indicate the seed to use for random numbers and the number of random numbers to print out.

Output

For each input you should output the first N numbers generated.

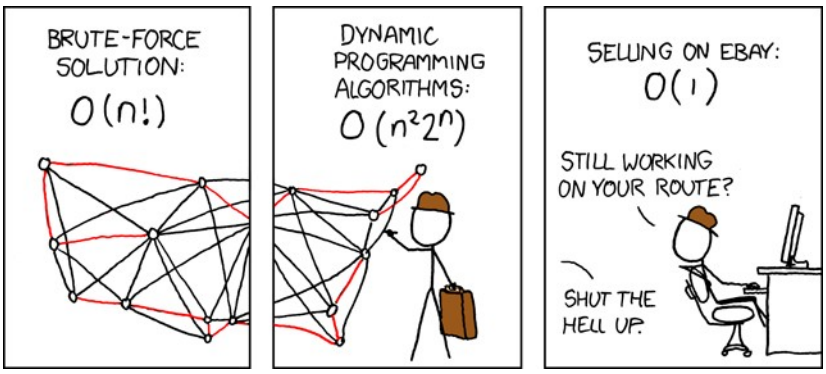
Sample Input

```
4
35 8
12 2
6 1
84 4
```

Sample Output

```
35 98 157 148 248 177 235 209
12 213
6
84 172 150 29
```

Problem 3 – The Maze-Traveling Salesman



Introduction

Last time you were adventuring in the Maze of Terror, you met a troupe of traveling salesmen. Since the maze is trapped, and they need to visit every shop once to sell their wares, they have trouble planning their own routes. The salesmen have asked you to make their travels more efficient by algorithmically finding the best route through the maze.

Input

The first line will contain the number of mazes (1<=I<=10). Each maze will be of size 10 cells long and 20 cells wide. Input will be given as the following symbols: % is the salesman's starting location, "0" is a wall, "-" is an open area you can walk through in one time unit, "*"is a trap that takes three time units to cross, and "1" is a shop that the salesman must visit. You can't go through walls. The salesman must visit every shop and return to the entrance. There will be no more than 10 shops in the maze. Note that you can walk through shops.

Output

For each maze, you will simply output the length of time it takes on the fastest route that visits every shop and returns to the origin. It will be possible to get to all the shops from the entrance.

Sample Input

```
2
--------------------
-0*00000-0-0000-----
-0--0--0-0-0--------
-000---0---0--000---
-0-0---0------------
-0-0---0------------
-0-00-0000000000100-
---0--------------*%
--0*0-0000000000100-
--------------------
--------------------
-1*0000000-0000-----
-0--0--0-0-0--0-----
-0000--0---0-00-----
```

```
-0-00--0-100-1------
-0-00--0---0--------
-1-00-0000000000000-
---0--------------*-
-00*000000000000000%
--------------------
```

Sample Output

```
14
72
```

Problem 4 – The Game: You Just Lost It

Introduction

Let me take a few moments to explain to you The Game:

1) The object of The Game is to forget about The Game.
2) Every time you remember The Game, you must say aloud, "I just lost The Game."
3) You are now playing the game. There is no getting out of it.

Your friends all play the game, and they are tired of losing all of the time. So, they've asked you to write a program that will tell them if a given line of text will make them lose it or not before they read it. (Really, by running the program, they have already lost The Game, but at least they won't actively do it.)

Input

The first line will contain the number of lines of text (1<=I<=50). The next input will be N lines each of which you will read in.

Output

For each input, if the string contained the words "The Game" in either upper or lower case, print "You lose". If it does not, print "You win".

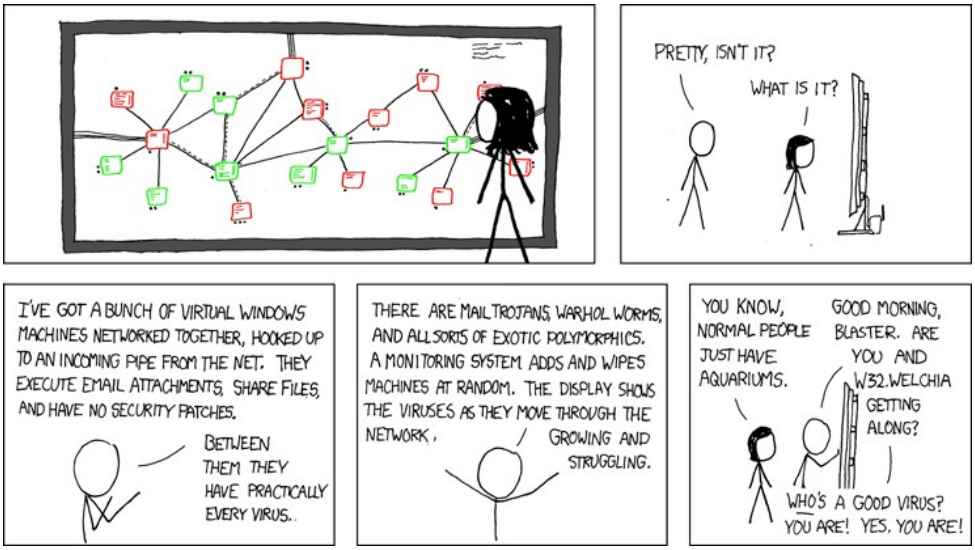
Sample Input

5
I just won the game.
You just lost the game.
XKCD is awesome.
Marble Cake. Also, The Game.
I just lost that chess game.

Sample Output

You lose
You lose
You win
You lose
You win

Problem 5 – The Little Virus That Could



Introduction

You recently joined a group of malicious hackers who have taken to writing efficient viruses for fun. Your job is to code the distribution algorithm for their latest virus, which will only bother infecting computers that can help spread it to other computers. That is, it will only infect a computer that is uninfected and still connected to another uninfected computer. Each infected computer will replicate to any attached, uninfected computer that is attached to another uninfected computer. You will always start with the first computer infected. You should determine is a computer was infected for the purposes of spreading by its state at the beginning of each round.

Input

The first line will contain the number of infected networks (1<=I<= 50). The next input will be the number of computers in the network (1<=N<=100). Next will follow N lines, which will have a list of 0s and 1s indicating which computers it connects to. (Note that a computer does not connect to itself and all input matrices will be symmetric.)

Output

For each input, output the total number of computers infected and the total time spent for the distribution.

Sample Input

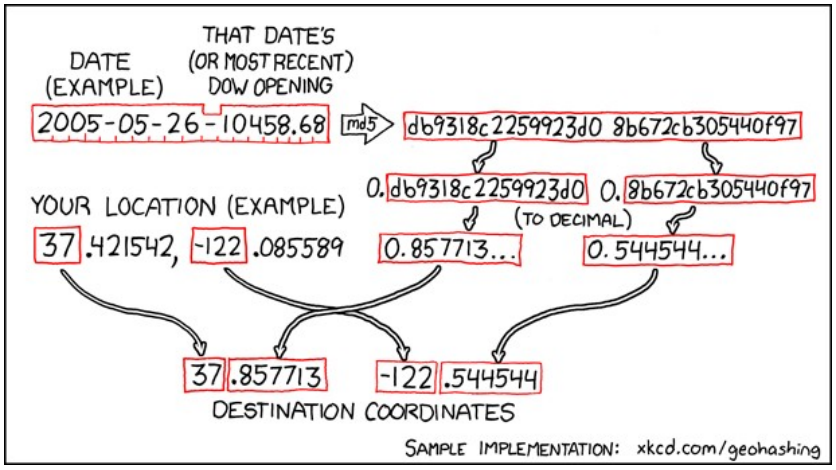
```
2
5
0 1 1 0 0
1 0 1 1 0
1 1 0 1 0
0 1 1 0 1
0 0 0 1 0
5
0 1 0 0 0
```

```
1 0 1 0 0
0 1 0 1 0
0 0 1 0 1
0 0 0 1 0
```

Sample Output

```
2 4
3 4
```

Problem 6 – I'm Late for a Very Important Date



Introduction

You are a huge fan of XKCD, and so you've decided to explore their interest in geohashing. So, you are going to write a small application to convert MD5 sums to GPS coordinates. However, unlike the comic, you are not going to use your location at all, but instead take the first two digits of each part of the hash and use those as the two digits before the decimal.

Input

The first line will contain the number of inputs, I (1 <= I <= 50). The next input will be N lines, each containing a single MD5 hash in hexadecimal.

Output

For each input, output the GPS coordinates it corresponds to, rounded off to 4 trailing digits.

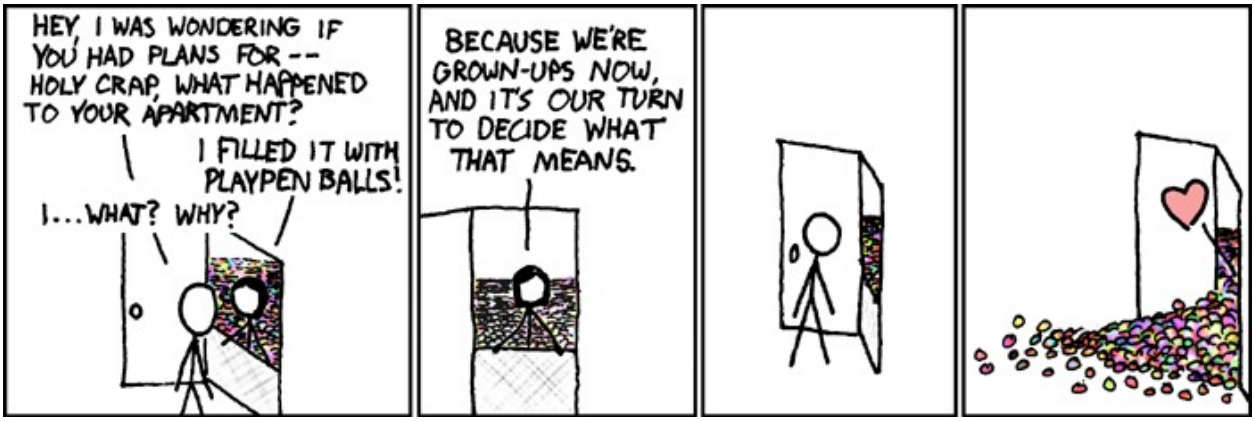
Sample Input

```
2
Db9318c2259923d0 8b672cb305440f97
19574A56B4837E95 DD39E636A8548B2F
```

Sample Output

```
219.5746 139.4030
25.3410 221.2262
```

Problem 7 – A Whole Apartment?



Introduction

Imagine trying to fill your apartment with playpen balls. A cubic foot of playpen balls costs $15.00, and you want to know how much it would cost to fill your whole apartment with them. You can only buy balls per cubic foot. You can't buy a fraction of a cubic foot.

Input

The first line will contain the number of inputs (1<=N<=50). After that will be N lines, each line containing S, the square footage of your apartment and H, the height you want the balls. (0.1<=S<=100,000, 0.1<=H<=10)

Output

For each input, output the total cost of filling the apartment in playpen balls. Use normal presentation of money with two digits after the decimal.

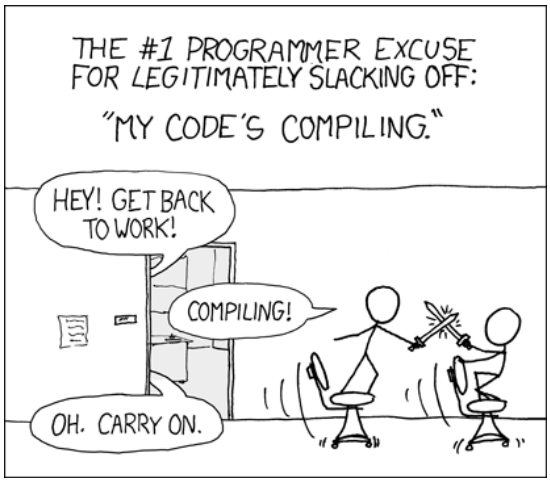
Sample Input

```
2
835.2 3.5
500.0 4.3
```

Sample Output

```
$43860.00
$32250.00
```

Problem 8 – The Compiling Problem



Introduction

You are just about to start a compile on the code for a large project. That means it is time to have some fun. The only question is, how much time do you have? For this problem you will write a little program to help you estimate that. By careful analysis of compiling times you have figured out roughly how the compile time scales with line counts, including variations from different types of lines. Use this information to calculate the total compile times for projects given different line counts.

The following table gives you compile times in milliseconds as a function of line count for different types of statements. The total compile time is the sum of all the separate times.

Line Type	**Time in milliseconds**
Assignment	10*A
Conditional	$100*C^2$
Loop	$150*L^2$
Method Call	$50*M^3$

Input

The first line will contain the number of inputs (1<=N<=50). After that will be N lines, each line containing the number of different types of lines as the values A, C, L, and M as space separated integers. (0<=10,000<=A, 0<=C<=1000, 0<=L<=1000, 0<=M<=100)

Output

For each input set, print the total compile time in milliseconds.

Sample Input

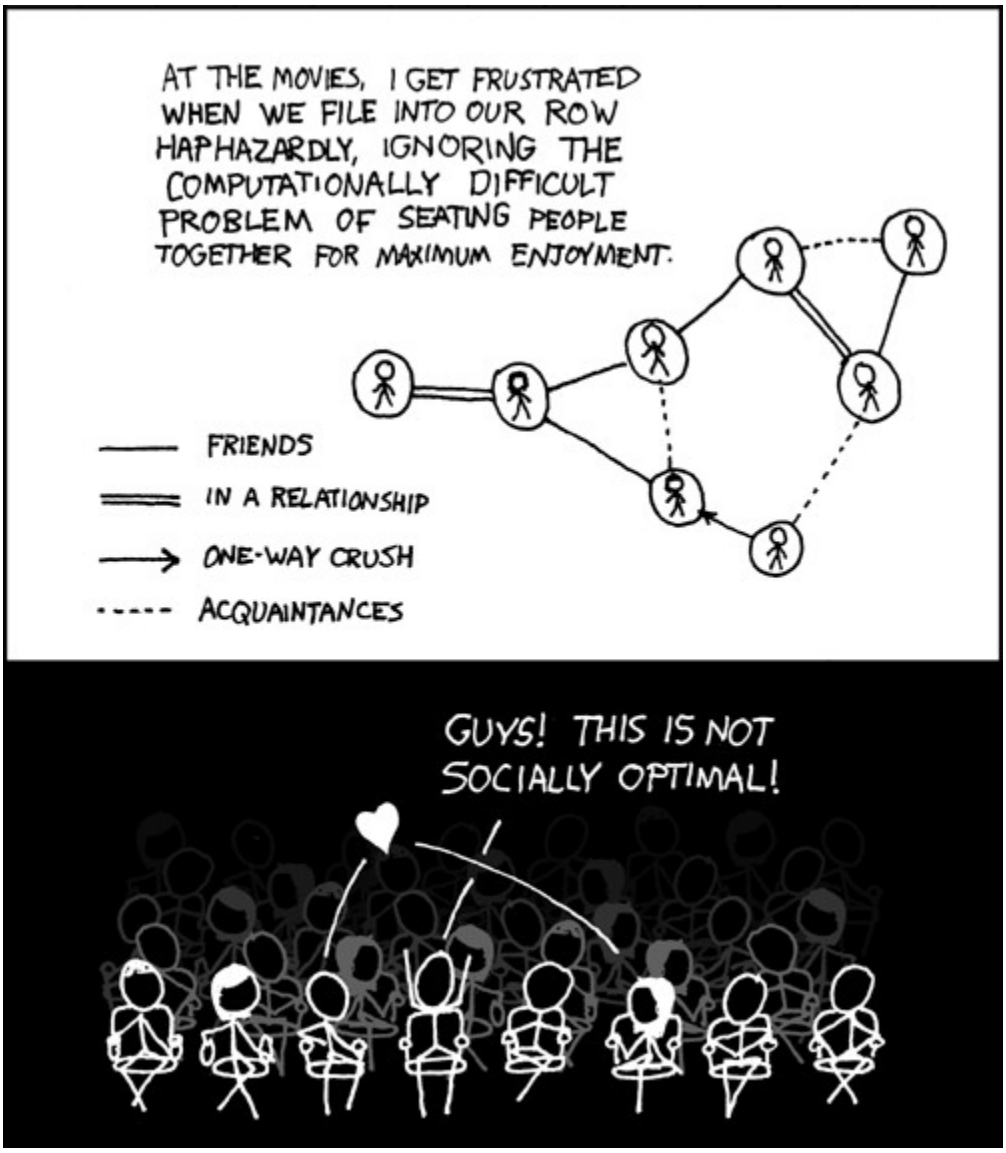
```
3
1000 0 0 0
200 50 20 10
```

```
2000 100 80 30
```

Sample Output

```
10000
362000
3330000
```

Problem 9 – Optimal Seating



Introduction

The cartoon above really describes this problem better than any words could. The basic idea is that you are given the various connections between a set of people. You need to find the idea sitting order and the "score" for that order. A seating order gets points for each connection for which people are sitting adjacent to one another. Different types of connections get different point values. Relationships are worth 10 points, one way crushes are worth 5, friendships are 3, and acquaintances are worth 1. People sitting next to one another who have no connection are worth zero points.

Input

The first line will contain the number of inputs sets (1<=I<=50). For each set there will be a line that has the names of all the people separated by commas (there can be spaces before or after the commas and names can have multiple words). After that is a line with a number (0<=C<=100) giving the number of connections between people. That is followed by C lines where each line has a connection.

The connections are given as two names and a type of connection. The values are comma separated, like the original names. The connection type will be one of the following: relationship, crush, friendship, or acquaintance. There will not be more than 16 people. The order of names in a connection does not matter. There will never be more than one type of relationship between any two people.

Output

For each input set, you will output one line. That line will give the score for the best arrangement, followed by a comma separated list of names specifying the arrangement. If there is a tie between arrangements for score, you should pick the one that is first alphabetically when all the names are put in the comma separated list.

Sample Input

```
2
John Doe, Jane Clark, Jeff L. Robinson
2
John Doe, Jane Clark, relationship
John Doe, Jeff L. Robinson, friendship
Boris, Annie, Katarina, Fred
4
Annie, Katarina, friendship
Boris, Fred, acquaintance
Boris , Annie ,  acquaintance
Katarina, Boris, relationship
```

Sample Output

```
13 Jane Clark, John Doe, Jeff L. Robinson
14 Annie, Katarina, Boris, Fred
```