# Colombian Collegiate Programming League

# CCPL 2017

Round 11 – October 7

## Problems

This set contains 10 problems; pages 1 to 18.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

# A - Circles

*Source file name:* `circles.c`, `circles.cpp`, `circles.java`, *or* `circles.py`

Enjoying a casual afternoon walk in the coordinate system, little Luka has encountered $N$ unique circles with its centers lying on the x-axis. The circles do not intersect, but they can touch (from the inside and the outside). Fascinated with circles, Luka wondered how many regions the circles divide the plane into. Of course, you are going to help him answer this question. A region is a set of points such that each two points can be connected with a continuous curve, without cutting through any of the circles.

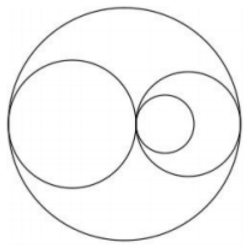


Figure 1: One of the possible layout of circles.

## Input

The input consists of several test cases and ends with end of file (EOF).

The first line of a test case contains the integer $N$ ($1 \leq N \leq 300000$), the number of circles. Each of the following $N$ lines contains two integers $x_i$ and $r_i$ ($-10^9 \leq x_i \leq 10^9$, $1 \leq r_i \leq 10^9$), the number $x_i$ representing the x coordinate of the $i^{th}$ circle and the number $r_i$ representing the radius of the $i^{th}$ circle. All the circles in each test case will be unique.

There is no blank line between test cases.

*The input must be read from standard input.*

## Output

Print one line for each test case: the required number from the task.

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| 2<br>1 3<br>5 1<br>3<br>2 2<br>1 1<br>3 1<br>4<br>7 5<br>-9 11<br>11 9<br>0 20 | 3<br>5<br>6 |

# B - String Game

*Source file name:* `string.c`, `string.cpp`, `string.java`, *or* `string.py`

Alice and Bob are playing the following game with strings of letters.

Before the game begins, an initial string and a target string are decided. The initial string is at least as long as the target string. Then, Alice and Bob take turns, starting with the initial string. Bob goes first. In each turn, the current player removes either the first or the last letter of the current string. Once the length of the current string becomes equal to the length of the target string, the game stops. If the string at the end of the game is equal to the target string, Alice wins the game; otherwise Bob wins.

Determine who will win the game if both players are playing optimally.

## Input

The input consists of several test cases. Each test case starts with $N$, the number of inputs to process. Each input consists of one line, which contains the initial string, followed by a space, followed by the target string. Each string consists of only lowercase letters. The total input length, for each case, will be less than 500000 characters.

*The input must be read from standard input.*

## Output

For each input, output the winner, which will either be Alice or Bob.

*The output must be written to standard output.*

| **Sample Input** | **Sample Output** |
|---|---|
| `5`<br>`aba b`<br>`bab b`<br>`aaab aab`<br>`xyz mnk`<br>`xyz xyz` | `Alice`<br>`Alice`<br>`Bob`<br>`Bob`<br>`Alice` |

## C - Mailing Origami

*Source file name:* `origami.c`, `origami.cpp`, `origami.java`, *or* `origami.py`

You would like to mail some origami you have made to your mom.

The price of mailing is dependent on the area of the envelope used to mail it: the smaller the envelope area, the less cost to ship. You cannot fold the origami shape to make it smaller. Of course, the envelope you are shipping the origami in must be rectangular.

Consider the vertices which represent the points along the boundary of the paper in order, such that the edge of the paper may fold over itself. Given the verticies describing the origami shape, what is the area of the smallest envelope that you can use to mail the origami?

### Input

The first line contains the integer $N$ ($3 \leq N \leq 100000$) which is the number of verticies describing your origami. The next $N$ lines contain two integers, $x$ $y$, the $x$-coordinate and $y$-coordinate of that particular vertex ($0 \leq x \leq 10000000$; $0 \leq y \leq 10000000$). You should assume all verticies are distinct, and that there is no line which contains all verticies.

*The input must be read from standard input.*

### Output

Output the area of the smallest envelope that will contain the origami, rounded to the nearest integer. You can assume that no test case will have the area of the smallest envelope containing the given vertices have a fractional part between 0.49 and 0.51.

*The output must be written to standard output.*

| **Sample Input** | **Sample Output** |
|---|---|
| `6`<br>`4 9`<br>`8 13`<br>`8 9`<br>`0 13`<br>`4 0`<br>`0 3` | `104` |

## D - Font

*Source file name:* `font.c`, `font.cpp`, `font.java`, *or* `font.py`

Little Ivica got himself a summer job at a company that produces computer fonts. The branch of the company where Ivica works at specialises in testing computer fonts and Ivica's team is responsible of testing only lowercase letters of the English alphabet.

The letters are tested so that various sentences using those letters are printed out and then manually (more accurately, visually) checked if everything is arranged properly. Only sentences which contain all 26 lowercase letter of the English alphabet (a-z) are used for testing purposes. These sentences are called test sentences.

You've probably already assumed that Ivica's job is to find test sentences. He has a dictionary which consists of $N$ words and has to calculate how many different test sentences can be made out of those words. Every word from the dictionary can be used only once in the sentence and the order of the words in the sentence is irrelevant (i.e. "uvijek jedem sarmu" and "jedem sarmu uvijek" are equal sentences).

### Input

The input consists of several test cases and ends with end of file (EOF).

The first line of a test case contains the integer $N$ ($1 \leq N \leq 25$), the number of words in the dictionary. Each of the following $N$ lines contains a single word from the dictionary, its length not exceeding 100. All the words from the dictionary are going to be unique.

There is no blank line between test cases.

*The input must be read from standard input.*

### Output

Print one line for each test case: the required number from the task.

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| 9<br>the<br>quick<br>brown<br>fox<br>jumps<br>over<br>a<br>sleazy<br>dog<br>3<br>a<br>b<br>c<br>15<br>abcdefghijkl<br>bcdefghijklm<br>cdefghijklmn<br>defghijklmno<br>efghijklmnop<br>fghijklmnopq<br>ghijklmnopqr<br>hijklmnopqrs<br>ijklmnopqrst<br>jklmnopqrstu<br>klmnopqrstuv<br>lmnopqrstuvw<br>mnopqrstuvwx<br>nopqrstuvwxy<br>opqrstuvwxyz | 2<br>0<br>8189 |

## E - Silly Sort

*Source file name:* `silly.c`, `silly.cpp`, `silly.java`, *or* `silly.py`

Your younger brother has an assignment and needs some help. His teacher gave him a sequence of numbers to be sorted in ascending order. During the sorting process, the places of two numbers can be interchanged. Each interchange has a cost, which is the sum of the two numbers involved.

You must write a program that determines the minimal cost to sort the sequence of numbers.

### Input

The input file contains several test cases. Each test case consists of two lines. The first line contains a single integer $n$ ($n > 1$), representing the number of items to be sorted. The second line contains $n$ different integers (each positive and less than 1000), which are the numbers to be sorted. The input is terminated by a zero on a line by itself.

*The input must be read from standard input.*

### Output

For each test case, the output is a single line containing the test case number and the minimal cost of sorting the numbers in the test case. Place a blank line after the output of each test case.

*The output must be written to standard output.*

**Sample Input**

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

**Sample Output**

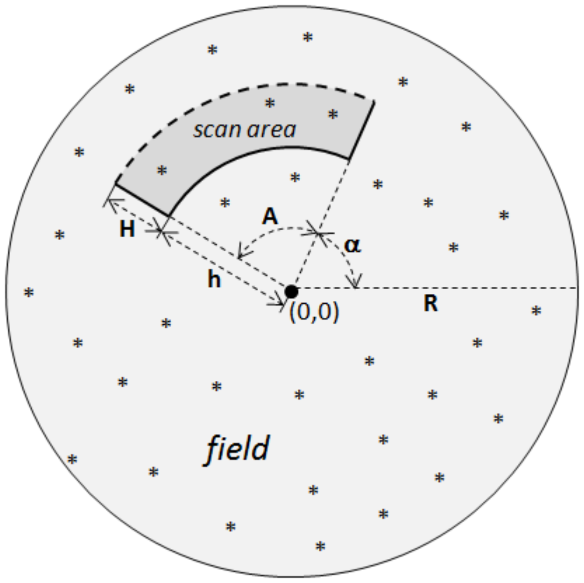
```
Case 1: 4
Case 2: 17
Case 3: 41
Case 4: 34
```

## F - Inspecting Radar

*Source file name:* `inspecting.c`, `inspecting.cpp`, `inspecting.java`, *or* `inspecting.py`

Radars Inc. is a worldwide renowned radar maker, whose excellent reputation lies on strict quality assurance procedures and a large variety of radar models that fit all budgets. The company hired you to develop a detailed *inspection* that consists of a sequence of $E$ *experiments* on a specific *surveillance model*.

There is a *field* represented with a polar coordinate plane that contains $N$ objects placed at positions with integer polar coordinates. The inspected model is located at the origin $(0, 0)$ of the field and can detect objects at a distance less than its *detection range* $R$ through a *scan area* defined by four adjustment parameters $\alpha$, $A$, $h$, and $H$, whose meaning is illustrated with the following figure:



Formally, the *scan area* of the model is the region described by the set of polar points

$$\{(r, \theta) | \, h \leq r < h+H, \ \alpha \leq \theta \leq \alpha+A\}$$

$\alpha$, $A$, $h$ and $H$ are four integer values where:

- $\alpha$ specifies the *start angle* of the radar's scan area ($0 \leq \alpha < 360$);
- $A$ specifies the *opening angle* of the radar's scan area ($0 \leq A < 360$);
- $h$ gives the *internal radius* of the radar's scan area ($0 \leq h < R$); and
- $H$ gives the *height* of the radar's scan area ($1 \leq H \leq R$).

An object placed at $(r, \theta)$ will be displayed by the model if $h \leq r < h+H$ and $\alpha \leq \theta \leq \alpha+A$, where the last inequality should be understood modulo 360 (i.e., adding and comparing angles in a circle).

Given $N$ objects placed on the field, you must develop an inspection of the surveillance model through the implementation of $E$ experiments with specific parameterizations. For each experiment you have to find the maximal number of objects on the field that the radar should display if the parameters $\alpha$ ($0 \leq \alpha < 360$) and $h$ ($0 \leq h < R$) are free to set (as integer numbers), and the parameters $H$ ($1 \leq H \leq R$) and $A$ ($0 \leq A < 360$) are given.

## Input

The input consists of several test cases. Each test case is described as follows:

- A line with two integer numbers $N$ and $R$ separated by blanks, representing (respectively) the number of objects located on the field and the detection range of the model ($1 \leq N \leq 10^4$, $2 \leq R \leq 10^2$).
- Each one of the following $N$ lines contains two integer numbers $r_i$ and $\theta_i$ separated by blanks, specifying the integer polar coordinates $(r_i, \theta_i)$ of the $i$-th object ($1 \leq r_i < R$, $0 \leq \theta_i < 360$, $1 \leq i \leq N$).
- The next line has an integer number $E$ indicating the number of experiments of the inspection ($1 \leq E \leq 10^2$).
- Each one of the following $E$ lines contains two integer numbers $H_j$ and $A_j$ separated by blanks, representing (respectively) the fixed height and the fixed opening angle that parameterize the $j$-th experiment ($1 \leq H_j \leq R$, $0 \leq A_j < 360$, $1 \leq j \leq E$).

For each test case you can suppose that there are not two different objects placed at the same integer polar coordinate. The last test case is followed by a line containing two zeros.

*The input must be read from standard input.*

## Output

For each test case of the input, print $E$ lines where the $j$-th line contains the maximal number of objects on the field that the radar should display according to the parameterization given for the $j$-th experiment ($1 \leq j \leq E$).

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| <pre>6 100<br>15 7<br>15 60<br>40 15<br>50 15<br>45 30<br>45 90<br>2<br>2 1<br>100 359<br>9 100<br>15 7<br>15 60<br>40 15<br>50 15<br>45 30<br>45 90<br>40 45<br>50 45<br>78 100<br>6<br>100 359<br>11 30<br>10 30<br>11 29<br>5 30<br>11 10<br>0 0</pre> | <pre>1<br>6<br>9<br>5<br>3<br>3<br>2<br>2</pre> |

## G - Division of Nlogonia

*Source file name:* `division.c`, `division.cpp`, `division.java`, *or* `division.py`

After centuries of hostilities and skirmishes between the four nations living in the land generally known as Nlogonia, and years of negotiations involving diplomats, politicians and the armed forces of all interested parties, with mediation by UN, NATO, G7 and SBC, it was at last agreed by all the way to end the dispute, dividing the land into four independent territories.

It was agreed that one point, called *division point*, with coordinates established in the negotiations, would define the country division, in the following way. Two lines, both containing the division point, one in the North-South direction and one in the East-West direction, would be drawn on the map, dividing the land into four new countries. Starting from the Western-most, Northern-most quadrant, in clockwise direction, the new countries will be called Northwestern Nlogonia, Northeastern Nlogonia, Southeastern Nlogonia and Southwestern Nlogonia.

The UN determined that a page in the Internet should exist so that the inhabitants could check in which of the countries their homes are. You have been hired to help implementing the system.

### Input

The input contains several test cases. The first line of a test case contains one integer $K$ indicating the number of queries that will be made ($0 < K \leq 10^3$). The second line of a test case contains two integers $N$ and $M$ representing the coordinates of the division point ($-10^4 < N, M < 10^4$). Each of the $K$ following lines contains two integers $X$ and $Y$ representing the coordinates of a residence ($-10^4 \leq X, Y \leq 10^4$).

The end of input is indicated by a line containing only the number zero

*The input must be read from standard input.*

### Output

For each test case in the input your program must print one line containing:

- the word `divisa` (means border in Portuguese) if the residence is on one of the border lines (North-South or East-West);
- `NO` (means NW in Portuguese) if the residence is in Northwestern Nlogonia;
- `NE` if the residence is in Northeastern Nlogonia;
- `SE` if the residence is in Southeastern Nlogonia;
- `SO` (means SW in Portuguese) if the residence is in Southwestern Nlogonia.

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| 3 | NE |
| 2 1 | divisa |
| 10 10 | NO |
| -10 1 | divisa |
| 0 33 | NE |
| 4 | SO |
| -1000 -1000 | SE |
| -1000 -1000 | |
| 0 0 | |
| -2000 -10000 | |
| -999 -1001 | |
| 0 | |

## H - Cargo Trains

*Source file name:* `trains.c`, `trains.cpp`, `trains.java`, *or* `trains.py`

The International Cargo and Packaging Company Inc. (ICPC Inc.) transports cargo between two cities: Source City and Sink City. ICPC Inc. does not have its own fleet, instead it contracts the service of two train companies, the company A and company B. Each company has its own network that connects some of the cities at prices that the company decides. For two given cities, it is possible that the route between them is served by both companies, only one company, or none. ICPC Inc. has reached an agreement with both companies that allows it to use their combined services at a discount price, but it has to follow these rules:

1. For a given shipment, ICPC Inc. must specify the percentage of participation of each company given by $a$ for company A and $(1 - a)$ for company B, for a given real number $a$ $(0 \leq a \leq 1)$.
2. When the segment between two cities is served by only one company, ICPC Inc. will pay the price corresponding to that company.
3. When the segment between two cities is served by both companies, the shipment can be shipped using a train from any of the two companies and will pay a fare equal to $a \times C_A + (1 - a) \times C_B$, where $C_A$ and $C_B$ correspond to the fares of company A and B respectively.
4. A shipment could pass through several intermediate cities. The total cost of the shipment corresponds to the sum of the costs of the individual segments between cities calculated according to rules 2 and 3.

ICPC Inc. needs your help to optimize its costs. Specificaly, ICPC Inc. needs to evaluate the cost of different alternatives that combine the participation of the two train companies in different proportions. Given the networks and prices of companies A and B, your task is to calculate the cost of $k$ different combination alternatives. Each alternative is specified by the participation of company A, which corresponds to a real number $0 \leq a \leq 1$.

### Input

The input contains multiple test cases. Each test case starts with a line with four integer values separated by spaces, $n$, $m_a$, $m_b$ and $k$, that correspond to the number of cities, the number of edges in the network of company A, the number of edges in the network of company B, and the number of combination alternatives respectively. The ranges for the values are: $2 \leq n \leq 100$, $1 \leq m_a, m_b \leq 5000$, and $1 \leq k \leq 10000$.

The next $m_a$ lines specify the network of company A. Each line has three integer values: $N_i$, $N_j$ and $C_{i,j}$, separated by spaces, with $0 \leq N_i, N_j < n$ and $0 \leq C_{i,j} \leq 1000000$. The network is an undirected graph and each edge is only listed once. $C_{i,j}$ corresponds to the cost of sending one kilogram of cargo from city $N_i$ to city $N_j$ or the other way around. Source City corresponds to 0 and Sink City to $n - 1$. Then next $m_b$ lines represent the graph corresponding to the network of company B represented in the same way as the network of company A. The last $k$ lines of the test case contain the different combinations to be evaluated, one combination per line. A combination is represented by a real number, $0 \leq a \leq 1$, with maximum 4 decimal digits. The value $a$ specifies company A's participation. Company B's participation is implicitely defined and corresponds to $1 - a$. You can suppose that there is at least one path between the Source City and the Sink City using routes served by any company.

The end of the input is indicated by the line
$-1\ -1\ -1\ -1$.

*The input must be read from standard input.*

## Output

For each combination alternative in each test case, the optimal cost of a trip from a city 0 to city $n - 1$ must be printed. If the answer has a decimal part, it has to be truncated without approximation.

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| <pre>3 2 2 3<br>0 1 100<br>1 2 200<br>0 1 200<br>1 2 150<br>0<br>1<br>0.5<br>-1 -1 -1 -1</pre> | <pre>350<br>300<br>325</pre> |

# I - Bus Tour

*Source file name:* `bus.c`, `bus.cpp`, `bus.java`, *or* `bus.py`

Imagine you are a tourist in Warsaw and have booked a bus tour to see some amazing attraction just outside of town. The bus first drives around town for a while (a *long* while, since Warsaw is a big city) picking up people at their respective hotels. It then proceeds to the amazing attraction, and after a few hours goes back into the city, again driving to each hotel, this time to drop people off.

For some reason, whenever you do this, your hotel is always the first to be visited for pickup, and the last to be visited for dropoff, meaning that you have to suffer through two not-so-amazing sightseeing tours of all the local hotels. This is clearly not what you want to do (unless for some reason you are *really* into hotels), so let's fix it. We will develop some software to enable the sightseeing company to route its bus tours more fairly—though it may sometimes mean longer total distance for everyone, but fair is fair, right?

For this problem, there is a starting location (the sightseeing company headquarters), $h$ hotels that need to be visited for pickups and dropoffs, and a destination location (the amazing attraction). We need to find a route that goes from the headquarters, through all the hotels, to the attraction, then back through all the hotels again (possibly in a different order), and finally back to the headquarters. In order to guarantee that none of the tourists (and, in particular, *you*) are forced to suffer through two full tours of the hotels, we require that every hotel that is visited among the first $\lfloor h = 2 \rfloor$ hotels on the way to the attraction is also visited among the first $\lfloor h = 2 \rfloor$ hotels on the way back. Subject to these restrictions, we would like to make the complete bus tour as short as possible. Note that these restrictions may force the bus to drive past a hotel without stopping there (this is not considered visiting) and then visit it later, as illustrated in the first sample input.

## Input

The first line of each test case consists of two integers $n$ and $m$ satisfying $3 \leq n \leq 20$ and $2 \leq m$, where $n$ is the number of locations (hotels, headquarters, attraction) and $m$ is the number of pairs of locations between which the bus can travel.

The $n$ different locations are numbered from 0 to $n-1$, where 0 is the headquarters, 1 through $n-2$ are the hotels, and $n-1$ is the attraction. Assume that there is at most one direct connection between any pair of locations and it is possible to travel from any location to any other location (but not necessarily directly).

Following the first line are $m$ lines, each containing three integers $u$, $v$, and $t$ such that $0 \leq u, v \leq n-1$, $u \neq v$, $1 \leq t \leq 3600$, indicating that the bus can go directly between locations $u$ and $v$ in $t$ seconds (in either direction).

*The input must be read from standard input.*

## Output

For each test case, display the case number and the time in seconds of the shortest possible tour.

*The output must be written to standard output.*

**Sample Input**

```
5 4
0 1 10
1 2 20
2 3 30
3 4 40
4 6
0 1 1
0 2 1
0 3 1
1 2 1
1 3 1
2 3 1
```

**Sample Output**

```
Case 1: 300
Case 2: 6
```

## J - Preferential Romance

*Source file name:* `romance.c`, `romance.cpp`, `romance.java`, *or* `romance.py`

*Marriage Success* (MS) is a marriage counseling service advising couples on how to improve the 'get along' experience. MS's idea is simple: each spouse writes down his/her preferences for various criteria of common interest. "Our criteria go beyond physical appearance and passion that guide early romance and tend to blind judgement. We want to understand your values as you live day by day. Happy couples are those whose preferences are compatible or can be made compatible."

Suppose $X$ and $Y$ are qualities to be considered. If a person declares that $X > Y$, it means that this person prefers quality $X$ to quality $Y$ (it does not mean that his/her mate should have a quality, it is only an opinion). Preferences are obviously irreflexive (i.e., $X \not> X$) and they are transitive (i.e., if $X > Y$ and $Y > Z$, then $X > Z$ –which can be abbreviated as $X > Y > Z$).

A couple is *fully compatible* if the preferences of the spouses are *consistent*, that is, if it is possible to arrange the qualities of interest of the spouses in a *compatibility list* reflecting both of their preferences. In this case, if a spouse says $X > Y$, qualities $X$ and $Y$ must occur in the compatibility list and moreover $X$ must be preferred over $Y$. If a couple is not fully compatible, then perhaps at least it is *passably compatible*: their preferences can be made consistent if some spouse drops at most one preference.

For example, newly-wed Alice and Bob declare their preferences with respect to the following qualities (that they observe in a possible mate): biker, cultured, enthusiastic, foodie, juggler, kayaker, movies, organized, puzzles, rich, theatre, and windsurfer. Their preferences are (observe that they could say nothing about qualities meaning that such quality does not have any importance):

Alice: organized > puzzles > rich, windsurfer > theatre, and rich > movies.

Bob: kayaker > movies > puzzles and rich > theatre.

In this case Alice and Bob are not fully compatible. To see that, a compatibility list should have rich before movies (Alice), movies before puzzles (Bob), and puzzles before rich (Alice), meaning that rich must occur before rich which is impossible. However, the couple is passably compatible: if Alice drops her preference rich > movies, then there is a compatibility list modeling both of their preferences:

kayaker > organized > movies > puzzles > rich > windsurfer > theatre.

MS needs a software solution to determine if clients are full compatible, passably compatible or none of these. Can you help?

### Input

The problem input consists of several test cases, each one defined by a set of lines establishing preferences of a couple. A test case is defined as follows:

- the first line contains two strings $A$ and $B$, separated by blanks, representing the name of the spouses,

- the second line is a sequence of strings of the form (one or more blanks separating items, including commas and final semicolon):

  $$q_{11}\ q_{12}\ \cdots\ q_{1r_1}\ ,\ q_{21}\ q_{22}\ \cdots\ q_{2r_2}\ ,\ \cdots\ ,\ q_{m1}\ q_{m2}\ \cdots\ q_{mr_m};$$

  meaning that person $A$ has sets of preferences ($q_{ij}$'s are strings denoting qualities):

  $$q_{11} > q_{12} > \ldots > q_{1r_1}, q_{21} > q_{22} \ldots > q_{2r_2}, \ldots, q_{m1} > q_{m2} > \ldots > q_{mr_m}$$

- the third line is of the form above representing the preferences of $B$.

Please consider that a quality name is a string with more than 0 and less than 11 characters, that couples may declare opinions about at most 100 different qualities, and that is guaranteed that the given data is well defined with respect to the above rules. Also, it is guaranteed that the information corresponding to each person does not include preference cycles (i.e., each person is self-compatible).

The end of the input is recognized by a line with $A = B = *$.

*The input must be read from standard input.*

## Output

For each given case, output one line with a single character `F`, `P`, or `N`, meaning that couple with spouses $A$ and $B$ is full compatible, passably compatible, or not compatible, respectively.

*The output must be written to standard output.*

| Sample Input | Sample Output |
|---|---|
| `Alice1 Bob1`<br>`organized puzzles rich , windsurfer theatre , rich movies ;`<br>`kayaker movies puzzles , rich theatre ;`<br>`Alice2 Bob2`<br>`organized puzzles rich , windsurfer theatre ;`<br>`kayaker movies puzzles , rich theatre ;`<br>`Alice3 Bob3`<br>`young busy rich , wallet tennis , rich movies , busy toys ;`<br>`toys movies busy , rich tennis busy , rich movies ;`<br>`* *` | `P`<br>`F`<br>`N` |