|  |  |  |
| --- | --- | --- |
|  | Trung ương hội Olympiad Tin học Châu Âu năm 2007  Ngày 1 - 7 |  |

**1.BỘ**

|  |  |
| --- | --- |
| ***Thời gian giới hạn:***  ***Giới hạn bộ nhớ:*** | 2 s  64 MB |

**CÔNG VIỆC**

Bạn được cho cơ cấu tổ chức của bộ. Mỗi đơn vị có duy nhất một cấp trên chính thức và nhiều nhất là 3 cấp dưới (có thê là không có ai). Trường hợp ngoại lệ duy nhất là bộ trưởng không có cấp cao hơn ( nhưng anh vẫn bị giới hạn là chỉ có nhiều nhất 3 cấp dưới). Không có thứ tự cụ thể của bất cứ cá nhân cấp dưới chính thức nào.

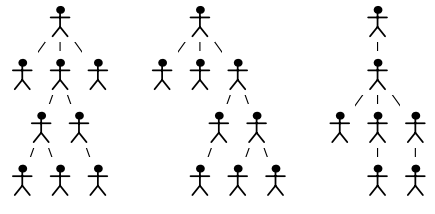
Sở bao gồm các đơn vị, các cấp dưới và cứ như thế. Có hai trường hợp đặc biệt trong sở là: Bộ đầy đủ (bắt đầu với bộ trưởng) và phòng ban một người chỉ bao gồm một nhân viên không có cấp dưới nào.

Chiều sâu của một sở chính là chiều dài của của chuỗi dài nhất x1, x2,…, xd của những đơn vị trong sở với xi là cấp trên của xi+1 với mỗi 1≤ i < d. Dễ thấy chiều sâu của một phòng ban một người là 1.

Hai sở A và B có cùng cơ cấu nếu mỗi đơn vị x của A tương ứng với một đơn vị x’ duy nhất của B và ngược lại mỗi đơn vị x’ của B tương ứng với một đơn vị x duy nhất của A. Đặc biệt, tất cả các đơn vị x và y đều phải tuân theo nguyên tắc sau đây: x là cấp trên của y khi và chỉ khi x’ (đơn vị tương ứng của x) là cấp trên

của y’ (đơn vị tương ứng của y). Dễ thấy nếu sở A và sở B có cùng cơ cấu thì người đứng đầu của sở A tương ứng với người đứng đầu của sở B và cả hai sở có cùng số nhân viên và chiều sâu.

Theo như hình minh hoạ sau, sở A và sở B có cùng cơ cấu, còn sở C có cơ cấu khác sở A và B.



Cơ quan A Cơ quan B Cơ quan C

Nhiệm vụ của bạn là để xác định số lượng các phòng ban với cấu trúc khác nhau cho tất cả chiều sâu.

Nói cách khác, bạn phải tạo ra một chuỗi n1, ..., nd mà d là độ sâu của cả bộ mà d là chiều sâu của bộ với mỗi i Bộ chứa chính xác ni phòng ban có chiều sâu i với cặp cấu trúc khác nhau.

**INPUT**

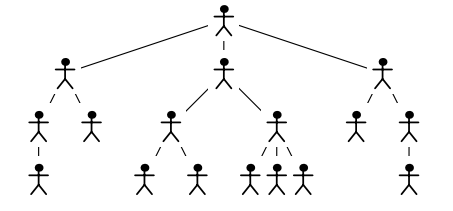
Đầu vào bao gồm một dòng duy nhất mô tả cơ cấu tổ chức của bộ gồm các thông số sau. Mỗi phòng ban được biểu thị dưới dạng (x1,…, xk) với 1≤ k ≤ 3 là số lượng cấp dưới của nhân viên cấp cao và của phòng ban xi’. Phòng ban một người được biểu thị bằng (). Cấu trúc của bộ được miêu tả bằng một đoạn mã của bộ đầy đủ.

Bộ không có quá 1 000 000 đơn vị.

**OUTPUT**

Đầu ra bao gồm d dòng, với d chỉ chiều sâu của bộ, có nghĩa là chiều sâu của phòng ban có người đứng đầu là Bộ trưởng. Dòng thứ i chỉ số lượng phòng ban có cấu trúc khác nhau ở độ sâu i.

**VÍ DỤ**



|  |  |
| --- | --- |
| **INPUT**  (((())())((()())(()()()))(()(()))) | **OUTPUT**  1  3  2  1 |

**2.PHÉP TÍNH XẤU XA**

|  |  |
| --- | --- |
| **Giới hạn thời gian:**  **Giới hạn bộ nhớ:** | 1 s  64 MB |

**CÔNG VIỆC**

Bởi vì số giá trị của x là vô cùng lớn mà đơn thức *f*(*x*) cũng chẳng ngắn gọn gì nên Joe đã nhờ bạn giúp. Cậu ta biết bạn là một lập trình viên tuyệt vời , vì thế cậu ta nghĩ bạn có thể dung máy tính để giải quyết vấn đề này. Để công việc của bạn dễ dàng hơn, Joe đã viết đơn thức *f* dưới dạng biểu thức ngược (sẽ được giải thích bên dưới), bởi vì cậu ta biết được rằng như vậy thì sẽ dễ dàng với máy tính hơn trong việc tính toán biểu thức. Nhiệm vụ của bạn là xác định chữ số cuối cùng của giá trị *f*(*x*) với đơn thức  *f* và các giá trị *x* cho trước trong hệ cơ số đếm B

**BIỂU THỨC NGƯỢC (Postfix notation)**

Biểu thức ngược (còn được biết đến với tên gọi Biểu thức Ba Lan ngược, RPN) là một cách khác để viết các biểu thức toán học. Sử dụng các kí hiệu thông dụng nhất trong các phép toán thường, các phép toán điều khiển biểu thức được đặt vào giữa các toán tử, ví dụ như 1+2, 1+2\*3, (1+2)\*3. Khônh giống như thế, trong biểu thức ngược, các phép toán điều khiển biểu thức được đặt ngay đằng sau toán các toán hạng: các ví dụ trên được biểu diễn dưới dạng 1 2 +, 1 2 3 + \*, 1 2 + 3\*.

Cách viết này trông thì có vẻ khó đọc với mắt thường (hay chí ít là cũng so với cách viết xuôi thông thường), tuy nhiên nó sẽ dễ hơn nhiều để viết một chương trình để tính toán biểu thức viết theo RPN so với viết một chương trình “hiểu” được biểu thức viết theo cách thông thường.

**HỆ CƠ SỐ ĐẾM B**

Các số nguyên thường biểu diễn dưới dạng số thập phân. Trong hệ đếm này, một số *dkdk-1…d1d0* được biểu diễn dưới dạng *dk10k+ dk-110k-1+…d1101+d0100*. Nếu số *10* được thay bằng một số nguyên B bất kì, *B ≥ 2*, và di là các số từ *0* đến *B-1* thì ta có được hệ đếm cơ số B. Giá trị của B là cơ số và di là các chữ số của hệ. Nói một cách chính xác hơn, trong hệ đếm cơ số B, giá trị *dkdk-1…d1d0* sẽ được biểu diễn dưới dạng *dkBk+ dk-1Bk-1+…d1B1+d0B0*. Tất nhiên, vì hệ đếm cơ số 10 của chúng ta chỉ có 10 chữ số nên các thiết lập của “chữ số” phải được nâng cao để biểu diễn các giá trị di > 9. Người ta đã quy ước các chữ số còn lại không thể biểu diễn trong hệ thập phân sẽ được biểu diễn bằng các chữ cái: *A* tương ứng với *10*, *B* tương ứng với *11*, v.v.Ví dụ số *29* trong hệ đếm cơ số 10 sẽ được biểu diễn trong hệ đếm cơ số 6 là *45*, trong hệ đếm cơ số 16 là *1D*.

**INPUT**

Dòng đầu tiên chỉ gồm duy nhất một số B và N. Trong đó, B là hệ cơ số đếm

(2≤ B ≤ 36), N là số giá trị của *x* (1≤ N ≤ 100 000).

Dòng thứ hai miêu tả biểu thức *f* theo cách biểu diễn ngược – dòng bao gồm nhiều phần tử cách nhau bởi một khoảng trắng. Các phần tử này có thể là:

* Một dãy các chữ số và chữ cái viết hoa.Dãy sẽ miêu tả các số được viết trong hệ cơ số đếm B như đã nói ở trên. Giả sử các giá trị này nhỏ hơn 2 000 000.
* Chữ cái viết thường x. Nó đại diện cho biến *x* trong đơn thức *f*
* Dấu +
* Dấu –
* Dấu \*

N dòng tiếp theo gồm một số hoăc một chữ cái viết hoa duy nhất thể hiện số *x* trong hệ đếm cơ số B tương ứng. Giả sử rằng các số nhỏ hơn 2 000 000.

Đảm bảo các dữ liệu đầu vào đều chính xác, ví dụ ở dòng thứ hai, đảm bảo nội dung của dòng thứ hai là một biểu thức hợp lệ viết theo kiểu biểu thức ngược và

mỗi dãy số và chữ cái trong dòng thứ hai là một số nguyên có giá trị trong hệ đếm cơ số B. Giả thiết rằng không có quá 100 000 kí tự trong dòng.

**OUTPUT**

Bao gồm chính xác N dòng. Dòng thứ *i* chỉ gồm duy nhất 1 kí tự, là một chữ số hoặc chữ cái viết hoa là chữ số cuối cùng của đơn thức *f*(*x*) (biểu diễn ở hệ cơ số B) với *x* cho bởi dòng thứ (*i+2*) của đầu vào.

**VÍ DỤ**

|  |  |
| --- | --- |
| **INPUT**  15 4  2 x \* 123A +  1  2  3  4 | **OUTPUT**  C  E  1  3 |

**3.NHỮNG MẢNH VỤN CỦA CÁNH BUỒM**

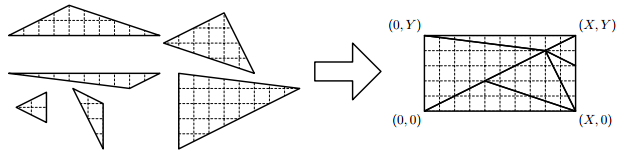
|  |  |
| --- | --- |
| ***File đầu vào:***  ***File dầu ra:***  ***Thời gian giới hạn:***  ***File nguồn*** | /mo/public/problem/sail/01.in, ..., 10.in  01.out, ..., 10.out  5h ☺  Không có! |

**CÔNG VIỆC**

Bạn được cho mô tả về các mảnh vụn hình tam giác của cánh buồm (bằng toạ độ của các đỉnh của mỗi tam giác trên mặt phẳng toạ độ) và kích thước ban đầu của cánh buồm vốn là một hình chữ nhật (xác định bởi độ dài hai cạnh X và Y). Nhiệm vụ của bạn là di chuyển các mảnh vụn (mà không được xoay chúng) để tái tạo lại cánh buồm ban đầu (xem hình). Đặc biệt, các tam giác phải được di chuyển sao cho:

* Toàn bộ diện tích tam giác được bao phủ
* Không có điểm nào cùng nằm trong hai tam giác khác nhau
* Không phần nào của tam giác nằm bên ngoài hình chữ nhật

Bạn có thể giả định rằng luôn tồn tại một giải pháp cho dữ liệu đầu vào. Đặc biệt, tổng diện tích các mảnh vụn hình tam giác bằng đúng với cánh buồm ban đầu. Có nhiều mảnh có thể có hình dạng giống nhau.



**INPUT**

Bạn không cần phải nộp chương trình giải quyết vấn đề đã đưa ra. Thay vào đó, trong thư mục /mo/public/problems/sail, bạn sẽ tìm thấy các file 01.in,…, 10.in.

Các tệp tin là các đầu vào của mỗi trường hợp thử nghiệm.

Mỗi file dữ liệu đầu vào (01.in,…, 10.in) miêu tả một trương hợp thử nghiệm duy nhất theo cấu trúc sau. Dòng đầu tiên gồm ba số X, Y, N cách nhau duy nhất một khoảng trắng. X và Y (1 ≤ X, Y ≤ 1 000 000) xác định kích thước của cánh buồm gốc, X là chiều rộng và Y là chiều dài, còn N (2 ≤ N ≤ 1000) là số mảnh vụn của cánh buồm.

N dòng tiếp theo mô tả một tam giác duy nhất: dòng thứ (*i+1*) bao gồm 6 số nguyên Ai,x, Ai,y, Bi,x, Bi,y, Ci,x, và Ci,y cách nhau bởi một dấu cách. Sáu số mô tả toạ độ 3 đỉnh Ai, Bi, Ci của tam giác: đỉnh Ai­ có toạ độ (Ai,x, Ai,y), đỉnh Bi có toạ độ

(Bi,x, Bi,y) và đỉnh Ci có toạ độ (Ci,x, Ci,y). Bạn có thể giả sử

0 ≤ Ai,x, Ai,y, Bi,x, Bi,y, Ci,x, Ci,y ≤ 1 000 000.

Lưu ý rằng vị trí của các tam giác là không thích hợp, ví dụ, trong mẫu , nếu thay dòng thứ 3 bằng chuỗi “6 6 14 5 16 6” thì kết quả không thay đổi.

**OUTPUT**

Với mỗi file đầu vào, bạn sẽ tạo ra mộ file đẩu ra tương ứng 01.out,…, 10.out.

Bạn không nên gửi bất cứ một tệp chương trình nào cho vấn đề này.

File đầu ra XX.out chỉ chứa chính xác N dòng mô tả thứ tự sắp xếp của các mảnh buồm sao cho thoả mãn cả 3 yêu cầu trong phần Công việc. Chính xác hơn, dòng thứ *i* sẽ chứa hai số nguyên A\*i,x và A\*i,y cách nhau duy nhất một dấu cách - mô tả toạ độ đỉnh Ai sau khi di chuyển mảnh vụn thứ i tới vị trí thích hợp. Lưu ý rằng các góc dưới cùng bên trái của cánh buồm luôn có tọa độ (0, 0) và góc trên bên phải có tọa độ (X, Y) như mô tả trong hình.

**VÍ DỤ**

Các dữ liệu đầu vào và đầu ra dưới đây tương ứng với hình ảnh minh hoạ bên trên (Lưu ý rằng các mảnh vụn bên trái chỉ cho biết hình dạng nhất định của từng mảnh chứ không cho biết vị trí của chúng)

|  |  |
| --- | --- |
| **INPUT**  6 10 5  4 0 12 4 4 5  4 5 12 4 14 5  0 3 10 3 4 5  4 5 10 3 8 7  7 7 7 10 5 11  0 1 2 0 2 2 | **OUTPUT**  0 0  0 5  0 0  4 2  10 0  8 4 |

**LƯU Ý: THỬ NGHIỆM CÁC GIẢI PHÁP CỦA BẠN TRƯỚC KHI TRÌNH**

Để giúp bạn tìm ra giải pháp, có một chương trình đơn giản, bạn có thể sử dụng để kiểm tra xem đầu ra của bạn là chính xác và hình dung các giải pháp. Chương trình này được gọi với hai tham số bằng cách sử dụng lệnh sau: draw\_sail XX.in XX.out với XX.in là một trong file đầu vào và XX.out là file đầu ra tương ứng. Chương trình này sẽ tạo ra và hiển thị một tập tin dưới dạng postscript miêu tả việc lắp ráp của những cánh buồm và kiểm tra lỗi. Đặc biệt, các khu vực của những cánh buồm được bao phủ bởi không có hình tam giác sẽ có màu đỏ, các khu vực được bao phủ bởi nhiều hơn một tam giác sẽ có màu xanh, và các bộ phận của hình tam giác nằm bên ngoài của những cánh buồm sẽ được màu xanh lá cây. Đừng quên rằng bạn có thể sử dụng phóng to / thu nhỏ chức năng của chương trình để hiển thị chi tiết của những cánh buồm. Bạn cũng có thể in trực tiếp các hình ảnh tương ứng với các tập tin đầu ra từ các chương trình.

**4.VÒNG CỔ**

|  |  |
| --- | --- |
| ***Thời gian giới hạn:***  ***Giới hạn bộ nhớ:*** | 3 s  128 MB |

**CÔNG VIỆC**

Nhiệm vụ của bạn là tạo một thư viện (thống nhất là bằng Pascal) để giúp các cuộc hội thoại kiểu này. Giao diện của thư viện sẽ được miêu tả trong phần tiếp theo; bạn có thể tìm thấy các thư viện mẫu trong các thư mục sau /mo/public/necklace/c, /mo/public/necklace/cpp, và/mo/public/necklace/pas (các thư mục con trong C và C++ cũng chứa các tệp tin tiêu đề necklace.h). Không có dữ liệu đầu vào hay dữ liệu đầu ra.

Thư viện của bạn sẽ làm việc với một tập hợp các day chuyền. Mỗi một chiếc vòng cổ là một chuỗi số nguyên lớn hơn 0 và nhỏ hơn 1 000 000, theo thứ tự từ trái sang phải. Mỗi một chiếc vòng cổ được xác định bởi một số nguyên không âm. Ban đầu, chỉ có chiếc vòng cổ mang thứ tự 0 là một chuỗi rỗng tượng trưng cho hiện tại.

Ban đầu, hệ thống đánh giá sẽ gọi thủ tục ***init*** một lần duy nhất. Sau đó, nó sẽ lien tục gọi theo một trật tự tuỳ ý các hàm ***create*** và ***pearl***. Tổng cộng, thư viện của bạn sẽ chạy ít nhất là 1 000 000 lần mỗi lần chạy thử.

Hàm ***create*** mỗi lần được gọi sẽ tạo ra một chiếc vòng cổ tương ứng với mỗi lần gọi. Số lượng vòng cổ sẽ nhiều hơn 1 đơn vị số lượng vòng cổ lớn nhất hiện có, tức là lần đầu tiên gọi hàm ***create*** tạo ra chiếc vòng mang số 1, lần gọi thứ hai tạo ra chiếc vòng mang số 2, v.v… Chiếc vòng mới sẽ được tạo nên dựa trên cơ sở của chiếc vòng ***from*** nhờ các hàm quy định bởi các thông số ***operation, on\_left*** và ***param***:

* Nếu tham số ***operation***  là chữ R (trong “remove” – loại bỏ) thì một số nguyên ở cuối chiếc vòng sẽ bị loại bỏ. Hàm ***param*** sẽ bị bỏ qua trong trường hợp này.
* Nếu tham số ***operation*** là chữ A (trong “add” – thêm) thì số nguyên ***param*** sẽ được thêm vào cuối chiếc vòng.

Nếu giá trị của ***on\_left*** là đúng (true) (trong C là khác 0) hàm sẽ thực hiện công việc của mình ở ngoài cùng bên trái của chiếc vòng cổ, ngược lại nó sẽ thực hiện công việc ở ngoài cùng bên phải của chiếc vòng cổ. Thông số ***from*** của hàm ***create*** được gọi luôn nhỏ hơn số thứ tự của chiếc vòng tạo nên bởi hàm lúc này, nghĩa là nó được tạo nên dựa trên một chiếc vòng trước đó. Bạn có thể giả định rằng sẽ không có bất cứ trường hợp nào yêu cầu loại bỏ một viên ngọc trai từ một dây chuyên rỗng.

Hàm ***pearl*** sẽ trả về giá trị của viên ngọc trai ngoài cùng bên trái nếu như ***on\_left*** trả về giá trị đúng (true) (khác 0 đối với C/C++), bằng không, nó sẽ trả về giá trị của viên ngọc trai ngoài cùng bên phải của chiếc vòng **neck\_id**. Hàm ***pearl*** sẽ chỉ đượ gọi bởi một chiếc vong cổ không rỗng đã được tạo ra bởi lần gọi hàm  ***create*** trước đó. Hàm này sẽ không làm thay đổi chiếc vòng cổ.

**MÔ TẢ GIAO DIỆN**

**Trong C/C++:**

*extern void init (void);*

*extern void create (int from, char operation, int on\_left, int param);*

*extern int pearl (int neck\_id, int on\_left);*

**Trong Pascal:**

*unit necklace;*

*interface*

*procedure init;*

*procedure create (from : longint; operation : char; on\_left : boolean; param : longint);*

*function pearl (neck\_id : longint; on\_left : boolean) : longint;*

Lưu ý rằng bạn có thể tạo các thủ tục bổ sung, chức năng, và các biến toàn cầu trong thư viện bạn chuẩn bị.

**VÍ DỤ**

Ví dụ sau đây cho thấy một trường hợp có thể xảy ra sau nhiều lần gọi hàm và giá trị trả về:

**Trong C/C++:**

*init ();*

*create (0, ’A’, 1, 5);*

*create (1, ’A’, 1, 3);*

*pearl (2, 0); /\* returns 5 \*/*

*create (2, ’R’, 0, 0);*

*pearl (3, 0); /\* returns 3 \*/*

*pearl (2, 0); /\* returns 5 \*/*

**Trong Pascal:**

*init;*

*create (0, ’A’, true, 5);*

*create (1, ’A’, true, 3);*

*pearl (2, false); { returns 5 }*

*create (2, ’R’, false, 0);*

*pearl (3, false); { returns 3 }*

*pearl (2, false); { returns 5 }*

**KIỂM TRA THƯ VIỆN CỦA BẠN**

Ngoài các thư viện mẫu bạn của bạn, thư mục con /mo/public/necklace chứa các file neck\_main.c, neck\_main.cpp, và neck\_main.pas là các file nguồn chính có thể biên dịch được với thư viện của bạn và sẽ kiểm tra những thư viện thông thường theo mẫu trên (chương trình sẽ cho bạn biết giá trị trả về của hàm ***pearl*** có chính xác hay không). Chương trình **compile** với tham số **necklace** sẽ biên dịch thư viện của bạn bằng các file nguồn chính (neck\_main.c, neck\_main.cpp hoặc neck\_main.pas) và kết quả trả về của chương trình **necklace** sẽ cho bạn kiểm tra thư viện của mình như mô tả trên1. Nếu chương trình chạy mà không có tham số, nó sẽ gọi thư viện của bạn như ví dụ và kiểm tra xem kết quả trả về của hàm ***pearl*** có chính xác hay không (nếu không chính xác, một tin nhắn sẽ được viết ở đầu ra

sai số chuẩn). Nếu chương trình được gọi với tham số *-i­* thì chương trình sẽ chuyển

sang chế độ tương tác. Chương trình sẽ đọc dữ liệu đầu vào và cho ra dữ liệu đầu ra tiêu chuẩn.

Trong chế độ tương tác, chương trình đầu tiên sẽ gọi thủ tục ***init*** và sau khi thủ tục này kết thúc, nó sẽ cho ra kết quả ***Initiated***. Sau đó, ba loại lệnh sau đây được nhập vào:

* **create *from operation on\_left param***

Chương trình khi thấy lệnh này sẽ gọi hàm ***create*** với các tham số ***from, operatio, on\_left, param***. Sau khi hàm kết thúc, chương trình xuất ra “**Done**”.

* **pearl *neck\_ ip on\_left***

Chương trình khi thấy lệnh này sẽ gọi hàm ***pearl*** với các tham số ***neck\_ id, on\_left***. Sau khi hàm kết thúc, chương trình xuất ra số “ Return X” với X là giá trị trả về của hàm .

* **quit**

Câu lệnh này sẽ kết thúc chương trình.

Lưu ý rằng các chương trình trong chế độ tương tác chỉ cung cấp cho bạn với câu trả lời của thư viện của bạn và nósẽ không kiểm tra xem các câu trả lời là đúng hay sai.

Sau đây là những câu lệnh và kết quả trả về khi chương trình đang ở chế độ tương tác tương ứng với những lệnh đến thư viện từ ví dụ:

**Trong C/C++:**

*Initiated*

*create 0 A 1 5*

1 Lưu ý rằng bạn có thể chạy chương trình **compile** với một tệp tin như một tham số điều sẽ cho phép bạn biên dịch thư viện với các tệp tin chính của riêng bạn

*Done*

*create 1 A 1 3*

*Done*

*pearl 2 0*

*Returns 5*

*create 2 R 0 0*

*Done*

*pearl 3 0*

*Returns 3*

*pearl 2 0*

*Returns 5*

*quit*

**Trong Pascal:**

*Initiated*

*create 0 A true 5*

*Done*

*create 1 A true 3*

*Done*

*pearl 2 false*

*Returns 5*

*create 2 R false 0*

*Done*

*pearl 3 false*

*Returns 3*

*pearl 2 false*

*Returns 5*

*Quit*

**5.BUỔI BIỂU DIỄN TẠI SÂN BAY**

|  |  |
| --- | --- |
| ***Thời gian giới hạn:***  ***Giới hạn bộ nhớ:*** | 5 s  64 MB |

**CÔNG VIỆC**

Nhiệm vụ của bạn là viết một chương trình kiểm tra xem các yêu cầu của hai màn biểu diễn có thể được sắp xếp theo một trật tự mà cả hai màn biểu diễn có thể cùng được tiếp tục. Với mỗi màn biểu diễn, bạn sẽ được cho một danh sách thứ tự hạ cánh và cất cánh của các đường bay. Thứ tự hạ cánh và cất cánh của mỗi màn trình diễn sẽ phải tuân the nguyên tắc dưới đây:

* Một đường băng sẽ không được đón máy bay cho đến khi nó để máy bay trước đó cất cánh.
* Chỉ có đường băng trước đó đón máy bay và chưa để máy bay đó cất cánh được quyền cho máy bay cất cánh.
* Tất cả các đường bay đều phải trống khi buổi quảng cáo kết thúc.

Mặt khác, nếu như có một đường băng cần thiết cho 1 buổi biểu diễn nào đó nhưng trước đó đang được dung để tiếp đón một máy bay khác thì màn biểu diễn đó có thể bị hoãn lại cho đến khi máy bay trong đường băng cất cánh. Tuy nhiên,

các màn biểu diễn có thể “khoá” cả sân bay nếu như một trong hai máy bay cần đường băng *A* để hạ cánh đồng thời đường băng B đã được đặt trước và máy bay còn lại cần đường băng *B* để hạ cánh nhưng đường băng *A* đã được đặt trước. Nếu điều này xảy ra, không một màn trình diễn nào có thể tiếp tục mà không vi phạm quy định hạ cánh và cất cánh.

Lưu ý rằng nhiệm vụ của bạn không phải là kiểm tra xem các màn biểu diễn có được sắp xếp sao cho sân bay không bị khoá.

**INPUT**

Dòng đầu tiên bao gồm một số nguyên *N* duy nhất là số đường băng của sân bay. Các dòng còn lại chia thành 2 khối, mỗi khối miêu tả một màn biểu diễn. Dòng đầu tiên của khối chỉ bao gồm một số nguyên *L* duy nhất, *2 ≤ L ≤ 5000* là tổng số lần cất cánh và hạ cánh của các máy bay xuyên suốt màn trình diễn. L dòng tiếp theo, mỗi dòng gồm một xâu có 3 kí tự *RES* hoặc *REL* và một số nguyên *A,*

*1≤ A ≤ N*, cách nhau một dấu cách duy nhất. Dòng bắt đầu với xâu *RES* đại diện cho máy bay hạ cánh và dòng bắt đầu với xâu *REL* đại diện cho máy bay cất cánh. Số *A* đại diện cho số thứ tự của đường băng có máy bay cất hoặc hạ cánh.

**OUTPUT**

Nếu các màn biểu diễn luôn kết thúc mà sân bay không bị “khoá” (kể cả có sự khác biệt về thời gian giữa các sân bay có máy bay hạ và cất cánh), kết quả đầu ra chỉ bao gồm một dòng duy nhất “***The performances will always finish.***”

Nếu như tồn tại một trường hợp hạ cánh và cất cánh dẫn đến toàn bộ sân bay bị khoá lại thì dữ liệu đầu ra chỉ gồm một dòng duy nhất mô tả trình tự dẫn đến khả năng này (lưu ý rằng có thể có nhiều khả năng xảy ra và bạn có thể chỉ ra một trường hợp tuỳ ý trong số chúng). Dãy mô tả trình tự của buổi diễn gồm một dãy các kí tự *1* và *2* (không có khoảng trắng giữa các số). Những số này đại diện cho thứ tự yêu cầu của các màn trình diễn (*1* cho màn trình diễn thứ nhất và *2* cho màn trình diễn thứ hai): đầu tiên, yêu cầu thứ nhất của màn trình diễn thứ *a1* được thực hiện, sau đó là yêu cầu thứ nhất của màn trình diễn thứ hai nếu như *a1 ≠ a2* và là yêu cầu thứ hai của màn trình diễn thứ nhất nếu như *a1 = a2*, v.v… Dãy phải là một chuỗi các hoạt động, tức là trong bất kì một thời điểm nào, mọi đường băng chỉ có thể tiếp nhận tối đa một máy bay. Vì dãy trên mô tả thứ tự dẫn đến việc sân bay bị khoá nên yêu cầu tiếp theo của màn trình diễn thứ nhất, cái mà chưa xuất hiện trong dãy nên yêu cầu một đường băng tiếp nhận yêu cầu của màn biểu diễn thứ hai, và yêu cầu tiếp theo của màn trình diễn thứ hai nên yêu cầu một đường băng có từ màn trình diễn thứ nhất.

**VÍ DỤ 1**

|  |  |
| --- | --- |
| ***INPUT***  2  4  RES 1  RES 2  REL 1  REL 2  4  RES 2  RES 1  REL 2  REL 1 | *Có 2 khả năng cho dữ liệu đầu vào như trên*  ***OUTPUT 1:*** 12  ***OUTPUT 2:*** 21 |

**VÍ DỤ 2**

|  |  |
| --- | --- |
| ***INPUT***  4  8  RES 1  RES 2  RES 3  REL 3  REL 2  RES 2  REL 1  REL 2  4  RES 3  REL 3  RES 4  REL 4 | ***OUTPUT:***  The performances will always finish. |

**6.KHO BẠC HOÀNG GIA**

|  |  |
| --- | --- |
| ***Thời gian giới hạn:***  ***Giới hạn bộ nhớ:*** | 0.5 s  64 MB |

**CÔNG VIỆC**

Người đứng đầu văn phòng thủ quỹ là George Skinflint. Mỗi nhân viên có 0, 1 hoặc nhiều hơn nữa các nhân viên cấp dưới và là nhân viên cấp dưới của một nhân viên duy nhất (ngoại trừ George Skinflint là nhân viên chịu trách nhiệm trước nhà vua). Số nhân viên không quá 1000. Nhiệm vụ của bạn là tìm xem số lượng tối đa các cặp có thể tạo thành bởi các nhân viên mà mỗi cặp được hình thành bởi một nhân viên và cấp dưới trực tiếp của mình. Ngoài ra, bạn cũng cần tìm xem có bao nhiêu cách hình thành số cặp như vậy. Lưu ý rằng có một vài nhân viên không cần phải được xếp vào một cặp.

**INPUT**

Dòng đầu tiên chỉ bao gồm một số N duy nhất là số nhân viên của văn phòng,

1 ≤ N ≤ 1000. Mỗi nhân viên được nhận diện bởi một số ID duy nhất trong khoảng từ 1 đến N. Số ID của nhân viên thủ quỹ (George) là 1. N dòng tiếp theo tương ứng với một nhân viên: Nó bao gồm số ID của nhân viên đó, số K các nhân viên cấp dưới trực tiếp của người đó, 0 ≤ K ≤ 999, và số ID của K nhân viên cấp dưới này cách nhau một dấu cách. Bạn có thể giả định rằng các dòng tương ứng của nhân viên không bao giờ xuất hiện trước cấp trên của mình.

**OUTPUT**

Dữ liệu đầu ra bao gồm 2 dòng. Dòng đầu tiên chỉ gồm một duy nhất một số M là số cặp nhiều nhất mà các nhân viên có thể tạo thành. Dòng thứ hai là số cách khác nhau tạo thành M cặp dựa trên luật của nhà vua.

**CÁCH CHẤM ĐIỂM**

Bạn có thể nhận được một phần số điểm của bài. Nếu như trong trường hợp bạn trả về đúng số lượng lớn nhất số cặp có thể tạo thành nhưng sai số cách khác nhau tạo thành số cặp hoặc ngược lại, bạn trả về sai số lần tạo thành số cặp lớn nhất nhưng trả về đúng số lượng cặp lớn nhất có thể tạo thành thì bạn sẽ nhận được 40% số điểm của bài kiểm tra đó.

**VÍ DỤ**

|  |  |
| --- | --- |
| **INPUT**  7  1 3 2 4 7  2 1 3  4 1 6  3 0  7 1 5  5 0  6 0 | **OUTPUT**  3  4 |