TIUP 2011

18 de Maio de 2011, 17:00 às 20:00

http://acm.ist.utl.pt/~mooshak/
acm@dei.ist.utl.pt

Problems

A - Consistency

B - Prize

C - The GO Game

D - Binary String Generators

E - Ball Game

Scientific Committee

- Alexandre Francisco
- Ana Teresa Freitas
- João Marques-Silva
- Luís Guerra e Silva
- Luís Russo
- Sara Madeira
- Vasco Manquinho

Available Compilers

Lang.	**Compiler**	**Version**	**Command Line**	**Ext.**
C	gcc	4.1.2	gcc -O -ansi -Wall -Wno-unused -lm $file	.c
C++	g++	4.1.2	g++ -O -ansi -Wall -Wno-unused -lm $file	.cpp
Java	jdk	1.6.0_02	javac $file	.java

A - Consistency

Problem

Some puzzles are defined by a set of statements and are known to be solved by logic deduction. In our case, the objective is not to solve a puzzle but to determine if a set of logic statements is consistent.

Consider a set of statements comparing the age between two persons. For example, one can say that John is older than Mary and that Mary is younger than Ann. However, a set of these statements can be inconsistent. For example, consider the following three statements:

1. John is older than Mary
2. John is younger than Ann
3. Ann is younger than Mary

Clearly, this set is inconsistent because from 1 and 2 one can infer that Ann must be older than Mary. However, that is not consistent with statement 3. Your goal is to make a software package that determines if a given set of statements comparing people's age is consistent.

Input

The input is a set of statements comparing two people's age. For easy parsing, the name of a person is of the form `pN` where N is an integer. The input format is as follows:

- One line with two integers S and P, where S represents the number of statements in the set and P denotes the number of different people used in the sentences.
- A list of S lines where each line contains a statement. Each statement has the following form:
  - The name of person1.
  - One symbol from the set $\{>, <\}$.
  - The name of person2.

  The symbol $>$ represents that person1 is older than person2. Otherwise, if the symbol $<$ is used, then person1 is younger than person2.

Limits

$0 < S \leq 10^5$
$0 < P \leq 10^5$

Output

The output is the word `Consistent` if the set of statements is consistent or `Inconsistent` if there is a contradiction in the set.

Sample Input 1

```
5 4
p1 > p2
p2 < p3
p1 > p3
p4 < p1
p4 > p3
```

Sample Output 1

```
Consistent
```

Sample Input 2

```
5 4
p1 > p2
p2 < p3
p1 > p3
p4 > p1
p4 < p3
```

Sample Output 2

```
Inconsistent
```

B - Prize

Problem

Congratulations! You are the 1000th customer at a store and you just won a prize. Your prize consists in any two items available in the store such that the sum of their values is not larger than V.

The store owner gives you the list of available items and their value. Your goal is to find two items that add up to the value V. If no two items add up to V, you must provide the solution that comes closer, such that their combined value is not larger than V.

The solution you provide consists of the two items and their combined value. In case of ties, you should report items that first appear in the list.

Input

The input format is as follows:

- One line with two integers I and V, where I denotes the number of available items and V denotes the maximum value of the combined items of the prize.
- A list of I lines where each line contains an item name and its value. The item name consists solely of letters and digits (maximum of 10 characters). The value is always a positive integer up to 10^6.

Limits

$2 \leq I \leq 10^5$
$0 < V \leq 10^6$

Output

The output consists of two lines. The first line is the combined value of the selected items and the second line contains the name of the selected items separated by a space. The selected items are sorted according to their order in the input list.

Sample Input 1

```
6 100
pants 50
socks 20
hat 30
shoes 50
jacket 100
shirt 50
```

Sample Output 1

```
100
pants shoes
```

Sample Input 2

```
7 120
tennis 100
ball1 45
ball2 30
clubs 85
shirt 45
racket 70
shorts 70
```

Sample Output 2

```
115
ball1 racket
```

C - The GO Game

Problem

Go is a territorial game. The board, marked with a grid of 19 lines by 19 lines, may be thought of as a piece of land to be shared between the two players. One player has a supply of black pieces, called stones, the other a supply of white. The game starts with an empty board and the players take turns, placing one stone at each turn on a vacant point. Black plays first, and the stones are placed on the intersections of the lines rather than in the squares. Once played, stones are not moved. However they may be surrounded and so captured, in which case they are removed from the board as prisoners.

Figure 1 presents a Go board.

Figure 1: A Go match

The objective of this problem is to determine which groups of stones are dead (i.e, have been captured).

The empty points on the board, which are horizontally and vertically adjacent to a stone, or a group of stones, are known as liberties. An isolated stone or group of stones is captured when enemy stones occupy all of its liberties. Stones occupying adjacent points constitute a connected group. It is important to remember that only stones that are horizontally or vertically adjacent are connected; diagonals do not count as connections.

Example of groups of stones that have been captured:

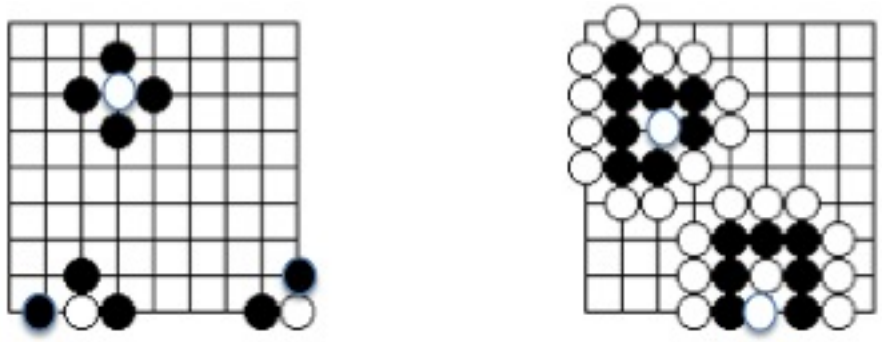

Figure 2: In figure a) the white stones are dead. In figure b) the black stones are dead since the whites did the last move.

Input

- A grid of 19 lines by 19 lines, corresponding to 361 interceptions. Each interception will be indentified with a . (if empty), a 0 (if has a white stone) and with 1 (if has a black stone).
- A number identifying the color of the stone that did the last move.

Output

The same board without the dead groups.

Sample Input 1

For simplicity, a grid of 9X9 was used in this example.

```
.........
...1.....
..101..11
...1...10
.......10
.11....10
0001....1
..11.....
.........
1
```

Sample Output 1

```
.........
...1.....
..1.1..11
...1...1.
.......1.
.11....1.
0001....1
..11.....
.........
```

D - Binary String Generators

Problem

A generator of binary strings (with symbols 0 and 1) can be defined recursively as follows:

1. Base case:
   For $b \in \{0, 1\}$, the rule $G \to b$, that rewrites G as b, defines G as a binary string generator.
2. Inductive step:
   If F and H are binary string generators, then the rule $G \to FH$, that rewrites G as F followed by H, defines G as a binary string generator.

A root binary string generator is defined from other generators. A binary string S is genrated by a root binary string generator, if S can be generated from R by applying rewriting operations of binary string generators.

Consider the following example:

$$\begin{aligned} R &\to AB \\ A &\to AZ \\ A &\to 0 \\ B &\to BO \\ B &\to 1 \\ Z &\to 0 \\ O &\to 1 \end{aligned}$$

The binary string 00111 is generated from R, but the binary string 010 is not. For example, 00111, can be generated as follows:

$$\begin{aligned} R &\to AB \to AZB \to 0ZB \to 00B \to 00BO \to \\ &\to 00BOO \to 001OO \to 0011O \to 00111 \end{aligned}$$

Given (i) a root generator of binary strings R; (ii) a set of binary string generators ; and (iii) a binary string S, the objective is to decide whether R generates S.

Input

The input is a sequence of rules of the form:

A. <id1><id2><id3>, denoting that <id1>generates <id2>followed by <id3>.

B. <id>0, denoting that <id>generates 0.

C. <id>1, denoting that <id>generates 1.

The first line denotes the root generator of binary strings, and must be of the form R<id2><id3>.

The rules are separated by a line with '#', which is then followed by a list of strings (on per line). For each string, the program must decide whether it is generated by R. The list of strings is terminated by a line with '#'.

Limits

There are at most 26 string generators, $A..Z$. The number of rewrite rules does not exceed 1024.

Output

The output must be 'yes' in case R generates a string S, and 'no' otherwise.

Sample Input 1

```
RAB
AAZ
A0
BB0
B1
Z0
01
#
00111
#
```

Sample Output 1

```
yes
```

Sample Input 2

```
RAB
AAA
A0
BBB
B1
#
110011
#
```

Sample Output 2

```
no
```

E - Ball Game

Problem

Portugal is a country with a long standing soccer tradition. Although the yellow and red leagues are over by now, we can still catch an orange league soccer game this Saturday. Some of the upcoming matches are Leixões vs Arouca, Freamunde vs Sta. Clara and Aves vs Feirense, among others. At this time we will be recovering from having programmed hard for a couple of days.

Coaches and players will be capitalizing on all the work they invested into training during the season. The games are usually balanced and the outcome is hard to guess. In this problem we will make an attempt to determine the winning team by using a simple model. Each team is given a score. If the score of team A is twice as big as the score of team B ($t_A > 2t_B$) then team A wins, and vice-versa. Otherwise, there is a tie. The score of a team depends on the team's ability to circulate the ball.

Depending on the players' positions and on their training, the players are assigned a passing score. This score is unidirectional meaning that passes from "Fonseca" to "Oliveira" may have a score of 30, in a given match, while passes from "Oliveira" to "Fonseca" may have a score of only 10. When the score for a pair of players is not given it is assumed to be 0. Still scores of 0 may be given explicitly. Moreover, players may also have a goal score that indicates their capability of scoring a goal. Likewise, when this score is omitted it is assumed to be 0.

The score of a team is calculated as the capability that the team has to circulate the ball, from its goal keeper to the other team's goal. Soccer is a collective sport, this means that we are not computing the shortest path, nor the one with the highest score, but the total capacity of the team to move the ball forward and into the other teams goal.

Input

The input contains the matches in a "Journada", each match contains a description of the competing teams.

- One line with the total number of matches J. For each of the matches the input contains a description of the two teams, as follows:
  - One line with the number of pair scores P followed by the team's name, with _ instead of spaces and no diacritics.
  - A sequence of P lines containing scores. Each line contains 3 numbers $P1$, $P2$ and S, where $P1$ is the number assigned to player 1, $P2$ is the number assigned to player 2, and S is the passing score between $P1$ and $P2$. Players are numbered from 1 to 11 and the goal keeper is always assigned with number 1. Moreover, the number $P2$ may be 0 and in this case S indicates the goal score of $P1$.

Limits

Each team has 11 players. The scores S are in $[0, 100]$ and the matches in a "Journada" are $J \leq 8$. The team name has a maximum of 80 characters.

Output

The output is a sequence of lines with the name of the winning team, or with the string “tie”.

Sample Input

```
2
18 Aves
1 2 1
2 3 1
3 4 2
4 5 3
5 6 4
6 7 5
7 8 6
8 9 7
9 10 8
10 11 9
11 0 10
2 1 11
1 3 12
3 5 13
5 7 14
7 9 15
9 11 16
11 1 17
18 Feirense
1 2 3
2 3 1
3 4 3
4 5 3
5 6 6
6 7 5
7 8 5
8 9 7
9 10 8
10 11 9
11 0 10
2 1 11
1 3 12
3 5 13
5 7 14
7 9 15
9 11 16
11 1 17
```

```
16 Covilha
1 2 6
2 3 6
3 4 6
4 5 6
5 6 1
5 7 1
5 8 1
5 9 1
5 10 1
5 11 1
6 0 1
7 0 1
8 0 1
9 0 1
10 0 1
11 0 1
16 Trofense
1 2 1
2 3 1
3 4 1
4 5 1
5 6 6
5 7 6
5 8 6
5 9 6
5 10 6
5 11 6
6 0 6
7 0 6
8 0 6
9 0 6
10 0 6
11 0 6
```

Sample Output

```
tie
Covilha
```