

## Practice Problem: Election

Source file: election.c, election.cpp, election.pas

### Introduction



very 25 years the people of Ruritania elect a new king. There are several candidates in the election, the candidate with the highest number of votes becomes the new king. Your job is to determine who has won the election. The vote counting committees has already counted the votes, you have to announce the next king.

### Input

The input contains several blocks of test cases. Each case begins with a line containing an integers  $1 \leq n \leq 1000$ , the number of candidates in the election. The next  $n$  lines correspond to the  $n$  candidates. Each line begins with an integer, the number of votes received by the candidate. This number of votes is followed by the name of the candidate (a single space separates the number and the name). The name is at most 10 characters long and contains only the 26 characters of the English alphabet. It can be assumed that there is no tie.

The input is terminated by a block with  $n = 0$ .

### Output

For each test case, you have to output a single line containing the name of the candidate with the highest number of votes.

#### Sample Input

```
3
42 Kuka
2000 Hapci
100 Szundi
5
1243 Li
4522 Hu
4987 Wu
931 Kim
174 Wong
0
```

#### Sample Output

```
Hapci
Wu
```

## Problem A: Clock

Source file: clock.c, clock.cpp, clock.pas

### Introduction



clocks are very important tools of modern life and technology: navigation, economy and travel cannot be imagined without them. However, the following problem has applications neither in navigation, economy nor travel. You are given a standard 12-hour clock with analog display, an hour hand and a minute hand. How many times does the minute hand pass the hour hand in a given time interval?

**Remark.** For the analogly-impaired here is how the standard clock works. There are 12 numbers on the perimeter of the display, both hands are in position 12 at noon and at midnight. The minute hand does a full turn in one hour, and the hour hand does a full turn in 12 hours.

### Input

The input file contains an indefinite number of lines. Each line contains an “initial time” and a “final time” (they can be anything between 00:00 and 23:59). No initial time and no final time will be an instant at which the minute hand just passes the hour hand. (In particular, 12:00 will not occur as an initial or final time.) The length of every interval is at most 24 hours, midnight may or may not be contained in the interval. The input file is terminated by a dummy line containing '00:00 00:00'.

### Output

For each line of the input (except the final dummy line), you have to output a line containing a single integer, the number of times the minute hand passes the hour hand in the given time interval.

#### Sample Input

```
12:50 13:02
03:08 03:30
23:00 03:20
01:02 12:50
00:00 00:00
```

#### Sample Output

```
0
1
4
11
```

## Problem B: Cities

Source file: cities.c, cities.cpp, cities.pas

### Introduction



ou have just arrived to the capital of Ruritania by plane. Two large ethnic groups live peacefully in this country: the Bandulus and the Tuluvus. They speak two completely different languages, therefore every newspaper, book, movie, etc. has a Bandulu and a Tuluvu version. Moreover, to make the situation more confusing, every city has a Bandulu and a Tuluvu name.

Your job is to determine the two names of every city in the country. With some effort, you managed to obtain two flight schedules, one in Bandulu and one in Tuluvu language. They told you that the two schedules contain the same flights. Every flight has a departure city, a destination city and a departure time (the flights are the same every day).

Reading the signs at the airport, you already know the name of the capital both in Bandulu and in Tuluvu language. It is known that every city in the country can be reached from the capital by plane (not necessarily by a direct flight). Moreover, take off is considered very complicated and dangerous, therefore two planes cannot depart from the same city in the same minute. Landing is much easier (after all, so far every plane managed to return to the land somehow), thus it is allowed that two planes arrive at the same time.

### Input

The input contains several blocks of test cases. Each case begins with a line containing two integers  $n$  and  $m$ . The number  $1 \leq n \leq 2000$  is the number of cities in the country and  $1 \leq m \leq 25000$  is the number of flights. The input is terminated by a block with  $n = m = 0$ . The second line of each block contains two words separated by a space: the name of the capital in Bandulu and in Tuluvu. City names are at most 10 characters long and contain only the 26 characters of the English alphabet. This is followed by  $2m$  lines, the first  $m$  lines contain the flight schedule in Bandulu and the next  $m$  lines contain the same schedule in Tuluvu. Each line begins with the time when the flight starts, followed by the departure and the destination city of the flight. The two schedules can contain the flights in different order: it cannot be assumed that the first flight in the Bandulu schedule is the same as the first flight in the Tuluvu schedule.

### Output

For each city, you have to output a line that contains first the Bandulu name and then the Tuluvu name. The cities should be ordered lexicographically by the Bandulu name. If it is sure that the two schedules are different (they cannot describe the same flights), then output the word 'inconsistent'.

#### Sample Input

```
4 4
Baand Tuulu
19:00 Baand Ndulu
12:30 Ndulu Bluuu
10:15 Bluuu Gooloo
12:30 Baand Gooloo
12:30 Tuulu Luvu
12:30 Vuuu Tvotvo
19:00 Tuulu Vuuu
10:15 Tvotvo Luvu
3 3
Baand Tuulu
12:00 Baand Bind
17:00 Bind Bund
22:30 Bund Bind
12:00 Tuulu Lulu
17:00 Lulu Vulu
22:30 Vulu Tuulu
0 0
```

#### Sample Output

```
Baand Tuulu
Bluuvu Tvotvo
Gooloo Luvu
Ndulu Vuuu
inconsistent
```

## Problem C: Rock stars

Source file: rockstars.c, rockstars.cpp, rockstars.pas

### Introduction



When a rock band gets very popular, then they are invited to lots of different places. However, every invitation cannot be accepted because the band cannot be in two different places at the same time, and traveling between cities requires nonzero time. The rock band 'Disaster Area' faces this situation: since they are extremely popular, they are invited to more events than they can possibly attend. Therefore they ask You to select some of these events in such a way that they have always enough time to travel to the next gig. Since you get 10% from the income of the band, you want to select the events in such a way that the band gets the most money.

The events are located in several cities. The band starts from city 1, on day 1 at 0:00. Rock stars use only the fastest and most comfortable form of traveling—teleportation. Sirius Cybernetics Corporation operates a teleportation service between the cities. You are given the list of teleportation connections offered by the company: the starting city, the destination city and the number of minutes the travel takes (in fact, the teleportation itself is instantaneous, but there are additional delays caused by filling out administrative forms, turning on the teleport equipment, etc.) You are also given the list of the events: the locations, the starting date, and how much money they pay for the performance. 'Disaster Area' performs the same program at every event, it is exactly 1 (one) minute long (believe us, it is long enough).

### Input

The input contains several blocks of test cases. Each case begins with a line containing three integers  $n$  ( $1 \leq n \leq 50$ ), the number of cities,  $m$  ( $1 \leq m \leq 2000$ ) the number of teleport connections between the cities, and  $k$  ( $1 \leq k \leq 10000$ ), the number of events where the band is invited to.

The next  $m$  line contains the description of the connection between the cities. Each line contains three positive integers  $x, y, t$  ( $1 \leq x, y \leq n, 1 \leq t \leq 1000$ ) with the meaning that from city  $x$  one can go to city  $y$  in  $t$  minutes. There is no fixed schedule, the band can use this connection any time it wants to. These connections are one directional only, so it does not necessarily mean that it is also possible to go from  $y$  to  $x$  in  $t$  minutes. It is possible that there is zero, or more than one direct connection from  $x$  to  $y$ .

The description of the connections is followed by  $k$  lines corresponding to the  $k$  events. The first integer on the line is the number of the city where the event takes place. The second integer is the day of the event, it is at most 365. This is followed by the starting time of the event in *hour:minute* form (the starting time can be anything between 0:00 and 23:59). The last integer on the line is the amount of money that the band gets for the performance ( $\leq 25000$ ).

The input is terminated by a block with  $n = m = k = 0$ .

### Output

For each test case you have to output a line containing a single integer, the maximum money that the band can get for the performances.

#### Sample Input

```
4 4 3
1 2 120
2 3 120
1 4 100
4 3 100
2 1 3:30 100
3 1 5:00 1000
4 1 2:00 50
4 4 3
1 2 120
2 3 120
1 4 100
4 3 100
2 1 2:45 100
3 1 5:00 1000
4 1 2:00 50
0 0 0
```

#### Sample Output

```
1050
1100
```

## Problem D: Winning Move

Source file: win.c, win.cpp, win.pas

### Introduction

Four by four tic-tac-toe is played on a board with four rows (numbered 0 to 3 from top to bottom) and four columns (numbered 0 to 3 from left to right). There are two players,  $\times$  and  $\circ$ , who move alternately with  $\times$  always going first. The game is won by the first player to get four of his or her pieces on the same row, column, or diagonal. If the board is full and neither player has won then the game is a draw.

Assuming that it is  $\times$ 's turn to move,  $\times$  is said to have a forced win if  $\times$  can make a move such that no matter what moves  $\circ$  makes for the rest of the game,  $\times$  can win. This does not necessarily mean that  $\times$  will win on the very next move, although that is a possibility. It means that  $\times$  has a winning strategy that will guarantee an eventual victory regardless of what  $\circ$  does.

Your job is to write a program that, given a partially-completed game with  $\times$  to move next, will determine whether  $\times$  has a forced win. You can assume that each player has made at least two moves, that the game has not already been won by either player, and that the board is not full.

### Input

The input file contains one or more test cases, followed by a line beginning with a dollar sign ('\$') that signals the end of the file. Each test case begins with a line containing a question mark ('?') and is followed by four lines representing the board; formatting is exactly as shown in the sample input. The characters used in a board description are the period (representing an empty space), lowercase  $x$ , and lowercase  $o$ .

### Output

For each test case, output a line containing the row and column position (separated by a space) of the first forced win for  $\times$ , or '#####' if there is no forced win.

For this problem, the first forced win is determined by board position, not the number of moves required for victory. Search for a forced win by examining positions (0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), ..., (3, 2), (3, 3), in that order, and output the first forced win you find. In the second test case below, note that  $x$  could win immediately by playing at (0, 3) or (2, 0), but playing at (0, 1) will still ensure victory (although it unnecessarily delays it), and position (0, 1) comes first.

### Sample Input

```
?
....
.xo.
.oX.
....
?
o...
.oX.
.xxx
xooo
$
```


### Sample Output

```
#####
0 1
```

## Problem E: Phone

Source file: phone.c, phone.cpp, phone.pas

### Introduction

he night life of OneDimensionVille is not very developed: the only place to go out is the disco of DJ Dedekind. Unfortunately, the disco burnt down after a wild party. This has the very sad consequence that there is no place left in the city where boys and girls can meet. The town council commissioned a special committee to find a solution to this problem. The committee proposed a very high-tech solution: building a video phone system would allow boys and girls to meet and chat virtually (it is only a coincidence that the owner of the largest video phone company, HyperVPhone, happens to be the nephew of the ex-wife of the brother-in-law of the head of the committee). The proposal was accepted and (what a surprise) HyperVPhone was chosen to build the video phone network. Since HyperVPhone receives a fixed amount of money for the project, it wants to build the cheapest network possible. Your task is to find this cheapest network.

There are exactly  $n$  boys and  $n$  girls in the city. HyperVPhone wants to build only  $n$  (bidirectional) connections: every boy will be connected to exactly one girl, and every girl will be connected to exactly one boy. The houses in OneDimensionVille are on one long street, there is unit distance between neighboring houses. The cost of building a connection between two houses is proportional to the *square* of the distance between the houses (if the distance is larger, then longer and more expensive cable is required). For example, connecting house 19 and 25 costs  $(19 - 25)^2 = 36$ . You have to connect the  $n$  boys with the  $n$  girls in such a way that minimizes the total cost of the connections.

### Input

The input contains several blocks of test cases. Each case begins with a line containing an integer  $n$  ( $1 \leq n \leq 5000$ ), the number of girls and boys in the city. The next  $n$  lines correspond to the  $n$  boys: each line contains a single integer ( $\leq 30000$ ), which is the number of the house where the boy lives. This is followed by  $n$  lines corresponding to the  $n$  girls. It can be assumed that at most one boy or girl lives in a house. The input is terminated by a block with  $n = 0$ .

### Output

For each test case, you have to output an integer, the minimum cost of the connections. It can be assumed that this number can be represented in a 32 bit integer.

#### Sample Input

```
2
1
10
16
12
3
15
2
16
12
8
11
0
```

#### Sample Output

```
157
68
```

## Problem F: Windows

Source file: windows.c, windows.cpp, windows.pas

### Introduction



The screen of monitors on computer systems are rectangles. The aspect ratio of a screen is its width divided by its height. This term can also be applied to rectangular windows that may appear on the monitor's screen, where it is defined as the width of the window divided by its height. For this problem we assume the dimensions of a monitor's screen and its windows are measured in integral numbers of pixels, the individual dots (arranged in a rectangular grid) that comprise an image.

Suppose your windowing software only allows windows to be resized in such a way that their aspect ratios are unmodified. For example, a window with a width of 150 pixels and a height of 100 pixels (and an aspect ratio of 150/100, or 1.5) can be resized so its width is 225 pixels and its height is 150 pixels (the aspect ratio remains unchanged, at 225/150, or 1.5), but a width of 224 and a height of 150 is not allowed, since that would change the aspect ratio. Each window can be moved to an arbitrary location on the screen, but the entire window must remain visible on the screen.

Given the size of a screen and the initial sizes of four different windows (as integer values for width and height), is it possible to resize (and relocate) the four windows so they completely cover the screen without overlapping each other? That's the question you are to answer in this problem.

For example, consider a square screen with four square windows. The aspect ratio for each of these is exactly 1. We are permitted to resize each of the four windows so they would completely fill the screen without overlapping. This case is illustrated by the first data set in the Sample Input, below.

### Input

The input contains several blocks of test cases. Each test case will contain five pairs of integers. The first pair ( $W_s, H_s$ ) specifies the width and height of the screen. The four remaining pairs ( $W_i, H_i$ , for  $i = 1$  to 4) specify the initial sizes of the windows. The input is terminated by a dummy line containing '0 0'. The numbers in the input are not greater than 1000.

### Output

Output should have one line for each test case. The line should contain the word 'Yes' if the screen can be completely covered by the (possibly resized and relocated) windows with no overlap, or 'No' if it cannot be so covered.

#### Sample Input

```
400 400 10 10 35 35 15 15 100 100
200 300 10 10 20 20 30 45 40 60
200 250 10 10 20 20 30 45 40 60
0 0
```


#### Sample Output

```
Yes
No
Yes
```

## Problem G: Boss

Source file: boss.c, boss.cpp, boss.pas

### Introduction

ogbert & Dogbert Inc. rents office space in several buildings throughout the city. Some of these offices are very cheap (for example, there is a building just beside a nuclear waste site), but the company rents some very expensive offices as well (with built-in swimming pool and plasma TV in every room). To reduce the cost of renting the offices, the chief efficiency analyst decided to move the employees into different buildings. However, not every employee can work in every building. There are employees that can work only where high-speed internet connection is available, some of them go to work by helicopter and require helicopter landing pads on the building etc. Furthermore, there is a very important rule that must be respected at all costs: under absolutely no circumstances should we put someone into the same building where his boss works. It is well known that the presence of the boss prevents any intelligent work being done. Your job is to move the employees into buildings such that these requirements are satisfied and the total cost is minimal.

### Input

The input contains several blocks of test cases. Each case begins with a line containing two integers  $n$  ( $1 \leq n \leq 4000$ ), the number of employees, and  $m$  ( $1 \leq m \leq 40$ ) the number buildings. The input is terminated by a block with  $n = m = 0$ . The second line of each case contains  $m$  positive integers (not greater than 2000) separated by spaces, the  $i$ th number is the cost of moving an employee into the  $i$ th building.

The next  $n$  lines corresponds to the  $n$  employees. Each line begins with the name of the employee, followed by the name of his boss. The names are at most 10 characters long and contain only the 26 characters of the English alphabet. The name of the boss is followed by a positive integer  $k$  (not greater than  $m$ ), which gives how many building is suitable for the employee. The rest of the line contains  $k$  positive integers, the buildings that are suitable for the employee.

Every employee at Dogbert & Dogbert Inc. has a boss, except the president. In the line corresponding to the president, the two names are the same. Clearly, there are no cycles in the hierarchy of the company. Note that an employee cannot work in the building where his boss works, but he can work where the boss of his boss works, only his direct boss has to be avoided.

### Output

For each test case, you have to output a line containing a single integer, the minium cost. If the employees cannot be assigned to the buildings in such a way that the requirements are satisfied, then output a line containing 'No solution.' instead.

### Sample Input

```
5 5
10 100 1000 55 1
Smith Balmer 1 2
Simonyi Gates 2 2 3
Gates Gates 3 3 1 5
Balmer Gates 2 1 5
Jackson Balmer 2 3 5
3 2
10 100
President President 2 1 2
Dogbert President 1 1
Dilbert President 1 2
0 0
```

### Sample Output

```
212
No solution.
```



## Problem H: Robots

Source file: robot.c, robot.cpp, robot.pas

### Introduction



he Restaurant at the End of the Universe is one of the most popular dining places in the galaxy. Every night, more and more people come to visit this famous place. To keep up with the increasing demand, the owner hired two intelligent robots, Marvin and Minsky, the latest models of Sirius Cybernetics Corporation. Their job is to go to the table of every customer and ask what he/she/it would like to eat/drink/photosynthesise/assimilate (depending on the life form). After taking the order, the robot sends a telepathic message to the kitchen with the requested items, and moves on to the next table. When the meal is ready, then the cook teleports it directly to the table of the customer.

Marvin and Minsky share the work, every customer is served by one of the two robots. Since they are very lazy and hate to walk (Sirius Cybernetics Corporation produces peculiar robots), they plan the work very carefully. Some of the customers are assigned to Marvin, the rest is assigned to Minsky, the goal is to find such an assignment that the robots have to walk as little as possible: the sum of the distances traveled by the two robots should be minimal. The robots must be fair to the customers assigned to them, they have to serve them on a First Come First Served basis. That is, if Mr. X arrives at 8:00 and Mr. Y arrives at 9:00, then Marvin cannot serve Mr. Y if he later serves Mr. X. However, it is possible that Marvin serves Mr. Y and later Minsky serves Mr. X: the order is important only between customers assigned to the same robot.

Given the position of the customers, your job is to find an optimal assignment and calculate the total distance walked by the robots. The robots walk on a straight line between the tables (even if there are other tables on this line, they don't care).

### Input

The input contains several blocks of test cases. Each case begins with a line containing an integer  $n$  ( $1 \leq n \leq 500$ ), the number of customers in the restaurant. The input is terminated by a block with  $n = 0$ . The first two lines of each block contain the initial position of the robots: the first line contains two integers  $x_1, y_1$  (separated by a space), the position of Marvin, the second line contains two integers  $x_2, y_2$ , the position of Minsky. This is followed by  $n$  lines, the  $i$ th line contains the  $x$  and  $y$  coordinates of the  $i$ th customer arriving. All coordinates in the input are nonnegative integers less than 2000.

### Output

For each test case, you have to output a line containing a single integer. This number is the total distance walked by the two robots, rounded down to the nearest integer.

#### Sample Input

```
2
100 200
200 200
0 200
100 300
4
0 0
1000 0
1100 0
1200 0
1300 0
100 0
0
```

#### Sample Output

```
241
400
```