Problem A. Team Arrangement

Barry Bennett, the coach of the Bings football team, wants to arrange his team for an important match against the Bangs. He decides on the formation he wants to play, for example 4-4-2, meaning that there will be four defenders, four midfielders, and two strikers (and of course, one goalkeeper). Your task is to determine the players who will play. For each available player, we know his role (e.g. midfielder). For each role, the players are selected in ascending order of their numbers. When the players are selected, you must determine the captain too, who is the player with the longest record in the team play. In case two players have the same record, the one with bigger number is chosen. Note that the captain is chosen among the players that are selected in the arrange.

Input (filename: A.in)

The input consists of multiple test cases. The first 22 lines of each test case contain the data for the 22 available players in this format: *number name role* $year_1$–$year'_1$ $year_2$–$year'_2$ ...
number is the player number (unique positive integer less than 100). *name* is a string of at most 20 letters. *role* is a single character among G, D, M, S, for goalkeeper, defender, midfielder, and striker respectively. Each $year_i$–$year'_i$ pair ($year_i \leq year'_i$) shows the player has been a member of the team between the specified years (inclusive). The years are in four-digit format. There is at least one and at most 20 such pairs, and the same year is not repeated more than once in the list. There is a 23rd line describing the desired formation, like 4-4-2 in that format. Note that there are only three numbers in the formation (so, 4-3-2-1 is not valid), none of them is zero, and their sum is always 10. The input is terminated by a line containing a single 0.

Output (Standard Output)

For each test case, output a list of 11 players chosen in the arrange. Each line must contain the player number, his name and his role, separated by single blank characters. The players must be sorted according to their role, in the order of goalkeeper, defenders, midfielders, and strikers. The players with the same role are sorted according to ascending order of their numbers. There is an exception that the captain always comes as the first player in the entire list. If it is not possible to arrange the team conforming to the desired formation, write a single line containing `IMPOSSIBLE TO ARRANGE` in the output. There should be a blank line after the output for each test case.

Sample Input and Output

A.in

```
9 PlayerA M 2000-2001 2003-2006
2 PlayerB M 2004-2006
10 PlayerC D 2001-2005
1 PlayerD D 2000-2001 2002-2004
11 PlayerE S 2003-2006
8 PlayerF M 2005-2006
22 PlayerG S 2005-2006
25 PlayerH G 2000-2001 2002-2003 2005-2006
6 PlayerI D 2003-2006
26 PlayerJ D 2003-2004 2000-2001
18 PlayerK M 2003-2004
19 PlayerL M 2000-2001 2003-2006
7 PlayerM S 2003-2006 1999-2001
21 PlayerN S 2003-2006
13 PlayerO S 2005-2006
15 PlayerP G 2001-2006
14 PlayerQ D 2003-2004
5 PlayerR S 2000-2005
20 PlayerS G 2000-2002 2003-2003
12 PlayerT M 2004-2005
3 PlayerU D 2000-2005
4 PlayerV M 2001-2004
4-4-2
0
```

Standard Output

```
7 PlayerM S
15 PlayerP G
1 PlayerD D
3 PlayerU D
6 PlayerI D
10 PlayerC D
2 PlayerB M
4 PlayerV M
8 PlayerF M
9 PlayerA M
5 PlayerR S
```

Problem B. Barbara Bennett's Wild Numbers

A *wild number* is a string containing digits and question marks (like `36?1?8`). A number *X* matches a wild number *W* if they have the same length, and every non-question mark character in *X* is equal to the character in the same position in *W* (it means that you can replace a question mark with any digit). For example, `365198` matches the wild number `36?1?8`, but `360199`, `361028`, or `36128` does not. Write a program that reads a wild number *W* and a number *X* from input, both of length *n*, and determines the number of *n*-digit numbers that match *W* and are greater than *X*.

Input (filename: B.in)

There are multiple test cases in the input. Each test case consists of two lines of the same length. The first line contains a wild number *W*, and the second line contains an integer number *X*. The length of input lines is between 1 and 10 characters. The last line of input contains a single character #.

Output (Standard Output)

For each test case, write a single line containing the number of *n*-digit numbers matching *W* and greater than *X* (*n* is the length of *W* and *X*).

Sample Input and Output

B.in	Standard Output
36?1?8 236428 8?3 910 ? 5 #	100 0 4

Problem C. Calculating Taxi Fare

The rules for calculating the taxi fares are quite complex. Many factors are to be considered in computing the taxi fares, including the length of the trip, the time of the day, the speed, etc. Every morning Bianca Bennett uses taxi to get to her office, she thinks if taximeters are programmed correctly. One day, she decided to write a program to calculate the taxi fares to check this.

Imagine a taxi passes through a sequence of streets S_1, S_2, ..., S_n in order. The length of S_i is L_i and it is assumed that the taxi travels in a constant speed and it takes M_i minutes to travel one kilometer in S_i. To make it simple, assume the passenger gets in at the start of a street S_i and gets out at the end of the destination street S_j (i.e., he does not get in or out in the middle of a street). The passenger is charged for each kilometer of the trip. The first ten kilometers of the trip cost 1000 Rials each. The next 20 kilometers (from 11 to 30) cost 250 Rials each. After that, each kilometer costs 100 Rials.

During the night, the fare is increased by 20%. The rule is that for each kilometer, if the taxi travels at least one minute during the time interval [12 AM, 6 AM], that kilometer will cost 20% more. Since driving in a heavy traffic costs more, if the average speed of the taxi is less than 30 km/h during the whole trip, the fare is increased by 10%.

Input (filename: C.in)

The input consists of multiple test cases. The first part of each test case is the sequence of streets the taxi travels. This comes in several lines, each describing one street in the form of *street-name length min*. *street-name* is a unique string of at most 20 letters and digits with no blank in it, and *length* and *min* are two positive integer numbers which are L_i (measured in kilometers, at most 200) and M_i (measured in minutes) respectively. Each street is visited once by the taxi. The first part of the test case is terminated by a line containing a single `$` character. The second part of the test case contains a single line of the form *source-street dest-street time*. The first two items are the names of the source and the destination streets respectively. The third item is the time the passenger gets in which is in standard 24-hours format (HH:MM). There is a line containing a single # character at the end of each test case. You may assume that the source and the destination streets belong to the input sequence of streets and the destination street does not come before the source street. The last line of the input contains two dash characters as shown in the sample input.

Output (Standard Output)

For each test case, output a line containing the fare of the passenger's trip.

Sample Input and Output

C.in	Standard Output
Khayyam 10 35 15thKhordad 50 15 Pamenar 15 40 $ Khayyam Pamenar 07:15 # Jenah 10 40 Nouri 50 70 Hemmat 30 25 Chamran 80 80 ValieAsr 30 20 $ Nouri ValieAsr 23:30 # --	21758 36432

Problem D. Party at Hali-Bula

Dear Contestant,

I'm going to have a party at my villa at Hali-Bula to celebrate my retirement from BCM. I wish I could invite all my co-workers, but imagine how an employee can enjoy a party when he finds his boss among the guests! So, I decide not to invite both an employee and his/her boss. The organizational hierarchy at BCM is such that nobody has more than one boss, and there is one and only one employee with no boss at all (the Big Boss)! Can I ask you to please write a program to determine the maximum number of guests so that no employee is invited when his/her boss is invited too? I've attached the list of employees and the organizational hierarchy of BCM.

Best,
--Brian Bennett

P.S. I would be very grateful if your program can indicate whether the list of people is uniquely determined if I choose to invite the maximum number of guests with that condition.

Input (filename: D.in)

The input consists of multiple test cases. Each test case is started with a line containing an integer n ($1 \le n \le 200$), the number of BCM employees. The next line contains the name of the Big Boss only. Each of the following n-1 lines contains the name of an employee together with the name of his/her boss. All names are strings of at least one and at most 100 letters and are separated by blanks. The last line of each test case contains a single 0.

Output (Standard Output)

For each test case, write a single line containing a number indicating the maximum number of guests that can be invited according to the required condition, and a word `Yes` or `No`, depending on whether the list of guests is unique in that case.

Sample Input and Output

D.in	Standard Output
6 Jason Jack Jason Joe Jack Jill Jason John Jack Jim Jill 2 Ming Cho Ming 0	4 Yes 1 No

Problem E. Against Mammoths

Back to year 3024, humans finally developed a new technology that enables them to conquer the alien races. The new technology made it possible to produce huge spaceships known as Saber Tooth spaceships as powerful as the aliens' defending mammoths. At that time, humans ruled several planets while some others were under control of the aliens. Using Saber Tooth ships, humans finally defeated aliens and this became the first Planet War in history. Our goal is to run a simulation of the ancient war to verify some historical hypotheses.

Producing each spaceship takes an amount of time which is constant for each planet but may vary among different planets. We call the number of spaceships each planet can produce in a year, the *production rate* of that planet. Note that each planet has a number of spaceships in it initially (before the simulation starts). The planets start producing ships when the simulation starts, so if a planet has n ships initially, and has the production rate p, it will have $n + p$ ships at the beginning of year 1, and $n + i \times p$ ships at the beginning of year i (years are started from zero).

Bradley Bennett, the commander in chief of the human armies, decided a strategy for the war. For each alien planet A, he chooses a corresponding human planet P, and produces spaceships in P until a certain moment at which he sends all spaceships in P to invade the planet A. No alien planet is invaded by two human planets and no human planet sends its spaceships to two different alien planets.

The defense power of the alien planets comes from their powerful mammoths. Each alien planet contains a number of mammoths initially and produces a number of mammoths each year (called the production rate of the planet). When a fight between spaceships and mammoths takes place, the side having the greater number of troops is the winner. If the spaceships win, the alien planet is defeated. In case the number of mammoths and spaceships are equal, the spaceships win.

The difficulty with planning this strategy is that it takes some time for the spaceships to reach the alien planets, and during this time, the aliens produce mammoths. The time required for spaceships to travel from each human planet to each alien planet is known. The ships can leave their planets only at the beginning of years (right after the ships are produced) and reach the alien planets at the beginning of years too (right after the mammoths are produced).

As an example, consider a human planet with two initial spaceships and production rate three invading an alien planet with two initial mammoths and production rate two. The time required to travel between the two planets is two years and the ships are ordered to leave at year one. In this case, five ships leave the human planet. When they reach the alien planet, they confront eight mammoths and will be defeated during the fight.

Bennett decided to prepare a plan that destroys every alien planet in the shortest possible time. Your task is to write a program to generate such a plan. The output is the shortest possible time (in years) in which every alien planet is defeated.

Input (filename: E.in)

There are multiple test cases in the input. The first line of each test case contains two numbers H and A which are the number of planets under the control of humans and aliens respectively (both between 1 and 250). The second line of the test case contains H non-negative integers $n_1\ m_1\ n_2\ m_2\ \dots\ n_H\ m_H$. The number n_i is the initial number of Saber Tooth spaceships in the i^{th} human planet and m_i is the production rate of that planet. The third line contains A non-negative integers which specify the initial number of mammoths and the production rate of the alien planets in the same format as the second line. After the third line, there are H lines each containing A positive integers. The j^{th} number on the i^{th} line shows how many years it takes a spaceship to travel from the i^{th} human planet to the j^{th} alien planet. The last line of the input contains two zero numbers. Every number in the input except H and A is between 0 and 40000.

Output (Standard Output)

The output for each test case contains a single integer which is the minimum time in which all alien planets can be defeated. If it is impossible to destroy all alien planets, the output should be `IMPOSSIBLE`.

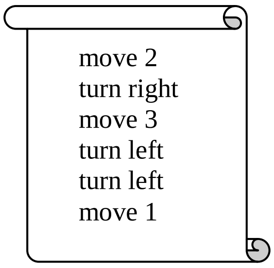
Sample Input and Output

E.in	Standard Output
2 1	6
2 3 0 3	
2 2	
2	
2	
0 0	

31st ACM International Collegiate Programming Contest, 2006-2007
Asia Region, Tehran Site
Sharif University of Technology, 14-15 Dec. 2006

Problem F. Chessboard Dance

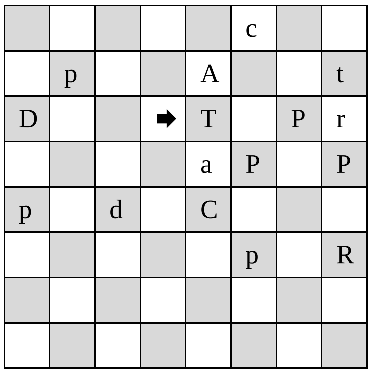
Another boring Friday afternoon, Betty the Beetle thinks how to amuse herself. She goes out of her hiding place to take a walk around the living room in Bennett's house. Mr. and Mrs. Bennett are out to the theatre and there is a chessboard on the table! "The best time to practice my chessboard dance," Betty thinks! She gets so excited that she does not note that there are some pieces left on the board and starts the practice session! She has a script showing her how to move on the chessboard. The script is a sequence like the following example:



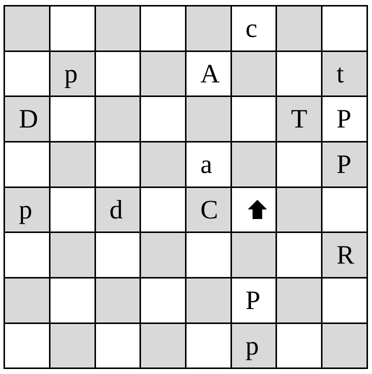
```
move 2
turn right
move 3
turn left
turn left
move 1
```

At each instant of time Betty, stands on a square of the chessboard, facing one of the four directions (up, down, left, right) when the board is viewed from the above. Performing a "move *n*" instruction, she moves *n* squares forward in her current direction. If moving *n* squares goes outside the board, she stays at the last square on the board and does not go out. There are three types of turns: turn right, turn left, and turn back, which change the direction of Betty. Note that turning does not change the position of Betty.

If Betty faces a chess piece when moving, she pushes that piece, together with all other pieces behind (a tough beetle she is!). This may cause some pieces fall of the edge of the chessboard, but she doesn't care! For example, in the following figure, the left board shows the initial state and the right board shows the state after performing the script in the above example. Upper-case and lower-case letters indicate the white and black pieces respectively. The arrow shows the position of Betty along with her direction. Note that during the first move, the black king (r) falls off the right edge of the board!



					c		
	p			A			t
D			➡	T		P	r
				a	P		P
p		d		C			
					p		R



					c		
	p			A			t
D						T	P
				a			P
p		d		C	⬆		
							R
					P		
					p		

You are to write a program that reads the initial state of the board as well as the practice dance script, and writes the final state of the board after the practice.

Input (filename: F.in)

There are multiple test cases in the input. Each test case has two parts: the initial state of the board and the script. The board comes in eight lines of eight characters. The letters `r`, `d`, `t`, `a`, `c`, `p` indicate black pieces, `R`, `D`, `T`, `A`, `C`, `P` indicate the white pieces and the period (dot) character indicates an empty square. The square from which Betty starts dancing is specified by one of the four characters `<`, `>`, `^`, and `v` which also indicates her initial direction (left, right, up, and down respectively). Note that the input is not necessarily a valid chess game status.

The script comes immediately after the board. It consists of several lines (between 0 and 1000). In each line, there is one instruction in one of the following formats (*n* is a non-negative integer number):

```
move n
turn left
turn right
turn back
```

At the end of each test case, there is a line containing a single # character. The last line of the input contains two dash characters.

Output (Standard Output)

The output for each test case should show the state of the board in the same format as the input. Write an empty line in the output after each board.

Sample Input and Output

F.in

```
.....c..
.p..A..t
D..>T.Pr
....aP.P
p.d.C...
.....p.R
........
........
move 2
turn right
move 3
turn left
turn left
move 1
#
--
```

Standard Output

```
.....c..
.p..A..t
D.....TP
....a..P
p.d.C^..
.......R
.....P..
.....p..
```

Problem G. Bribing FIPA

There is going to be a voting at FIPA (Fédération Internationale de Programmation Association) to determine the host of the next IPWC (International Programming World Cup). Benjamin Bennett, the delegation of Diamondland to FIPA, is trying to seek other delegation's support for a vote in favor of hosting IWPC in Diamondland. Ben is trying to buy the votes by diamond gifts. He has figured out the voting price of each and every country. However, he knows that there is no need to diamond-bribe every country, since there are small poor countries that take vote orders from their respected superpowers. So, if you bribe a country, you have gained the vote of any other country under its domination (both directly and via other countries domination). For example, if C is under domination of B, and B is under domination of A, one may get the vote of all three countries just by bribing A. Note that no country is under domination of more than one country, and the domination relationship makes no cycle. You are to help him, against a big diamond, by writing a program to find out the minimum number of diamonds needed such that at least *m* countries vote in favor of Diamondland. Since Diamondland is a candidate, it stands out of the voting process.

Input (filename: G.in)

The input consists of multiple test cases. Each test case starts with a line containing two integers *n* ($1 \le n \le 200$) and *m* ($0 \le m \le n$) which are the number of countries participating in the voting process, and the number of votes Diamondland needs. The next *n* lines, each describing one country, are of the following form:

CountryName DiamondCount $DCName_1$ $DCName_2$ ...

CountryName, the name of the country, is a string of at least one and at most 100 letters and *DiamondCount* is a positive integer which is the number of diamonds needed to get the vote of that country and all of the countries that their names come in the list $DCName_1$ $DCName_2$ ... which means they are under direct domination of that country. Note that it is possible that some countries do not have any other country under domination. The end of the input is marked by a single line containing a single # character.

Output (Standard Output)

For each test case, write a single line containing a number showing the minimum number of diamonds needed to gain the vote of at least *m* countries.

Sample Input and Output

G.in	Standard Output
3 2 Aland 10 Boland 20 Aland Coland 15 #	20

Problem H. Treasure of the Chimp Island

Bob Bennett, the young adventurer, has found the map to the treasure of the Chimp Island, where the ghost zombie pirate LeChimp, the infamous evil pirate of the Caribbeans has hidden somewhere inside the Zimbu Memorial Monument (ZM^2). ZM^2 is made up of a number of corridors forming a maze. To protect the treasure, LeChimp has placed a number of stone blocks inside the corridors to block the way to the treasure. The map shows the hardness of each stone block which determines how long it takes to destroy the block. ZM^2 has a number of gates on the boundary from which Bob can enter the corridors. Fortunately, there may be a pack of dynamites at some gates, so that if Bob enters from such a gate, he may take the pack with him. Each pack has a number of dynamites that can be used to destroy the stone blocks in a much shorter time. Once entered, Bob cannot exit ZM^2 and enter again, nor can he walk on the area of other gates (so, he cannot pick more than one pack of dynamites).

The hardness of the stone blocks is an integer between 1 and 9, showing the number of days required to destroy the block. We neglect the time required to travel inside the corridors. Using a dynamite, Bob can destroy a block almost immediately, so we can ignore the time required for it too. The problem is to find the minimum time at which Bob can reach the treasure. He may choose any gate he wants to enter ZM^2.

Input (filename: H.in)

The input consists of multiple test cases. Each test case contains the map of ZM^2 viewed from the above. The map is a rectangular matrix of characters. Bob can move in four directions up, down, left, and right, but cannot move diagonally. He cannot enter a location shown by asterisk characters (`*`), even using all his dynamites! The character (`$`) shows the location of the treasure. A digit character (between `1` and `9`) shows a stone block of hardness equal to the value of the digit. A hash sign (`#`) which can appear only on the boundary of the map indicates a gate without a dynamite pack. An uppercase letter on the boundary shows a gate with a pack of dynamites. The letter A shows there is one dynamite in the pack, B shows there are two dynamite in the pack and so on. All other characters on the boundary of the map are asterisks. Corridors are indicated by dots (`.`). There is a blank line after each test case. The width and the height of the map are at least 3 and at most 100 characters. The last line of the input contains two dash characters (`--`).

Output (Standard Output)

For each test case, write a single line containing a number showing the minimum number of days it takes Bob to reach the treasure, if possible. If the treasure is unreachable, write `IMPOSSIBLE`.

Sample Input and Output

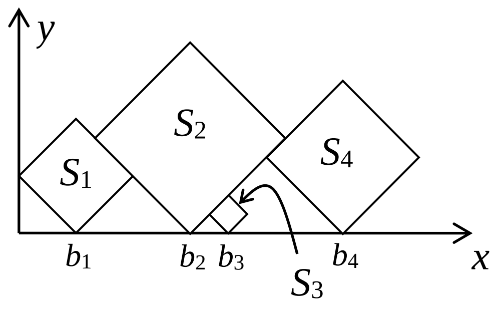
H.in	Standard Output

`*****#*********`
`*.1....4..$...*`
`*..***..2.....*`
`*..2..*****..2*`
`*..3..******37A`
`*****9..56....*`
`*.....******..*`
`***CA**********`
 	
`*****`
`*$3**`
`*.2**`
`***#*`
 	
`--`
	`1`
	`IMPOSSIBLE`

Problem I. Kadj Squares

In this problem, you are given a sequence S_1, S_2, ..., S_n of squares of different sizes. The sides of the squares are integer numbers. We locate the squares on the positive *x*-*y* quarter of the plane, such that their sides make 45 degrees with *x* and *y* axes, and one of their vertices are on *y*=0 line. Let b_i be the *x* coordinates of the bottom vertex of S_i. First, put S_1 such that its left vertex lies on *x*=0. Then, put S_i, ($i > 1$) at minimum b_i such that

a) $b_i > b_{i-1}$ and
b) the interior of S_i does not have intersection with the interior of $S_1...S_{i-1}$.



The goal is to find which squares are visible, either entirely or partially, when viewed from above. In the example above, the squares S_1, S_2, and S_4 have this property. More formally, S_i is visible from above if it contains a point *p*, such that no square other than S_i intersect the vertical half-line drawn from *p* upwards.

Input (filename: I.in)

The input consists of multiple test cases. The first line of each test case is *n* ($1 \le n \le 50$), the number of squares. The second line contains *n* integers between 1 to 30, where the i^{th} number is the length of the sides of S_i. The input is terminated by a line containing a zero number.

Output (Standard Output)

For each test case, output a single line containing the index of the visible squares in the input sequence, in ascending order, separated by blank characters.

Sample Input and Output

I.in	Standard Output

4
3 5 1 4
3
2 1 2
0
	1 2 4
	1 3