Problem A: Setting up a Football Team

Sharif is to put together a football team from the students who have volunteered to play in the team. A group of ex-players have set up some physical criteria for each position which are Maximum Age, Minimum Weight, and Minimum Strength. The criteria for each position in the team are as follows:

Position	Max. age	Min. Weight	Min. Strength
Mid-field	30	70	500
Forward	26	60	200
Defense	36	80	300

Using this information, you need to develop a program that reads in the physical attributes of several students and outputs what position(s) they are able to play.

Input (Standard Input)

In each line of the input, the age, weight and strength of a student are given as integer numbers (less than 1000); all separated with spaces. The input terminates with "0 0 0".

Output (Standard Output)

For each student, you must print one line listing the positions that student can play. A student can play a position if his weight and strength are not less than those specified in the above table and his age is not greater than that specified in the above table. If a student can play multiple positions, output them in the order listed above, separated by a space. If a student cannot play any position, print "No positions" on the line.

Sample Input and Output

Standard Input	Standard Output
27 85 350 24 65 630 23 75 550 37 210 500 0 0 0	Defense Forward Mid-field Forward No positions

Problem B: Fractals

Fractals are self-similar patterns. One simple example is to perform the following step on a string of dashes with length 3^k to produce similar patterns.

Replace the middle third of each piece of dashes with spaces. Repeat until each piece consists of a single dash.

For instance, if k is 3, we start with a string of 27 dashes:

Remove the middle third of the string:

--------- ---------

and remove the middle third of each piece:

--- --- --- ---

and again:

- - - - - - - -

The process stops when each group of dashes has length 1. You must write a program to specify whether i^{th} character in the output of the last step is "-" or not.

Input (Standard Input)

There are multiple test cases in the input. Each test case is given in a line containing two numbers k (at most 30) and i (at most 3^k). The goal is to find whether i^{th} character in the output of the last step of the above process working on a initial string of length 3^k is "-" or not. The input terminates with a line containing 0 0.

Output (Standard Output)

For each test case if the i^{th} character in the last output is "-" you must print "Yes"; otherwise print "No".

Sample Input and Output

Standard Input	Standard Output
3 6 3 7 0 0	No Yes

Problem C: Cutting a Cake

We have a large cake of size 100 × 100 × 1 cm. The surface of the cake can be viewed as a 100 × 100 grid, with each grid box being 1 × 1 cm. We want to cut the cake into pieces using a knife, initially located at a vertex of the grid. We are restricted to move the knife only along the edges of the grid, without lifting up the knife at any step. After finishing the cut, we can pick only those pieces of the cake which are cut completely from all sides. In other words, if the trajectory of the knife is given as a rectilinear curve along the grid edges, we can pick only those pieces which are enclosed by the trajectory curve. Note that if a piece has a side on the grid boundary, the piece can picked provided that all its sides even its side on the grid boundary are cut by the knife. We want to compute the amount of cake we can pick if the trajectory of the knife on the cake is given.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains three integers x, y and n, where (x, y) denotes the starting point of the knife on the grid, and n is the number of knife moves. (x, y, and n are all non-negative integers, x is the distance in centimeters from the left edge of the cake, y is the distance in centimeters from the bottom edge of the cake. x, y and n range from 0 to 100, inclusive.)
The next n lines will contain a character d and an integer i, separated by a space. The character d is from the set {'L', 'R', 'U', 'D'}, indicating the left, right, up, and down directions, respectively, and the integer i indicates how far in that direction the knife is moved.
You can assume that the knife trajectory will never leave the 100 × 100-surface of the cake. The trajectory may or may not be closed. It may cross itself, or may trace an edge several times even boundary edges. The input terminates with a line containing 0 0 0.

Output (Standard Output)

For each data set, output a single line containing the volume of cake in cubic centimeters that can be picked after the cut (i.e., enclosed by the knife trajectory).

Sample Input and Output

Standard Input	Standard Output
10 10 8 U 20 R 50 U 20 L 30 D 40 R 20 U 30 R 20 0 0 8 U 25 R 25 U 25 R 25 D 25 L 25 D 25 L 25 0 0 0	1000 1250

Problem D: Placing a disk inside a polygon

A simple polygon is defined as a flat shape consisting of straight, non-intersecting line segments that are joined pair-wise to form a closed path. A simple polygon is called convex if for every pair of points within the polygon, every point on the straight line segment that joins them is also within the polygon. A disk can be placed inside a simple polygon if its center can be placed inside the simple polygon in such a way that the whole disk lies inside the polygon. You are responsible to compute the largest disk can be placed inside a convex polygon.

Input (Standard Input)

There are multiple test cases. The first line of each test case contains a single number n (at most 100) which is the number of vertices. Then n lines coming: the i^{th} line contains the x and y coordinates of vertex v_i respectively separated with spaces. Both coordinates are real numbers with absolute value at most 10,000. The polygon is obtained by joining v_i to v_{i+1} for all i from 1 to n ($v_{n+1} = v_1$). The input terminates with a line containing 0.

Output (Standard Output)

For each test case, the output is just a line as described next. If the polygon is not convex, output "Non convex polygon". Otherwise, output the radius of the maximum disk can be placed inside the polygon with exactly two digits after the decimal point.

Sample Input and Output

Standard Input	Standard Output
5 1.0 1.0 2.0 2.0 1.75 2.0 1.0 3.0 0.0 2.0 5 1.0 1.0 2.0 2.0 1.75 2.5 1.0 3.0 0.0 2.0 0	Non convex polygon 0.71

Problem E: Hubs

As Bob is forgetful, he has a strange plan of driving around the city which reduces the number of routes he must memorize. He selects two distinct locations as hubs (H_1 and H_2) and partitions other locations into two groups. One group is assigned to H_1 while the other is assigned to H_2. He then memorizes the shortest route from each location to the hub associated with. If he wants to drive from A to B, he first drives from A to the hub associated with A, then to the other hub (if A and B are not in the same group), and then to B along the shortest route. Note that Bob always drives to hubs even if he visits the destination several times on the path. You should help Bob to find the best two hubs and the best partitioning which minimize the average distance of the trips between any two locations.

Input (Standard Input)

The input contains several test cases. The input of each test case is a weighted undirected graph which models the city map. The first line contains two numbers $2 \leq n \leq 100$, $1 \leq m \leq 1000$ which are the number of vertices (locations) and the number of edges (streets), respectively. Each of the next m line describe one edge of the graph with three numbers a, b and d where $1 \leq a \leq n$, $1 \leq b \leq n$ are location ids and $1 \leq d \leq 1000$ is the length of the street connecting a to b (all streets are bidirectional). There may be more than one street between a pair of locations, and a street may start and end at the same location. You may assume there exists a path between any two locations. The input terminates with 0 0.

Output (Standard Output)

For each test case, output the minimum average distance of the trips between any two distinct locations with exactly two digits after the decimal point.

Sample Input and Output

Standard Input	Standard Output
3 2 1 2 40 2 3 20 5 5 1 2 2 2 3 3 2 4 3 3 4 5 3 5 1 0 0	40.00 4.20

Problem F: Auctions

An auction is a process of buying and selling goods or services by offering them up for bid, taking bids, and then selling the good to the highest bidder. Precisely, first a minimum price is set for an item, then bidders propose their bids. At the auction end time of the item, it is sold to the bidder who proposed the maximum bid. You are to write a program that tracks auctions for several items and output the results.

Input (Standard Input)

Input consists of three sections namely items, bidders and bids coming respectively in the input.

Item section: The first line of the item section contains a single number $k < 100$ denoting the number of items. Each next k lines contain item id, its minimum price and its auction end time. Item id is a non-negative integer number, and minimum price is a non-negative real number, and auction end time is in the 24-hour format of XX:YY:ZZ where XX is in hours from 00 to 23, YY is in minutes from 00 to 59, and ZZ is in seconds from 00 to 59. Note a bid for an item is acceptable if it is proposed not later than the auction end time of the item and the proposed price is not less than the minimum price of the item.

Bidder section: This section starts with a line containing a single number $m < 100$ which is the number of bidders. Each next m lines contains bidder id and its balance account separated by spaces. Bidder id is a non-negative integer number, and balance account is a non-negative real number. Note that a bidder can propose a price larger than its account balance but at the auction end time this bid is considered as an invalid bid and is not taken into account. At the auction end time of an item, the account balance of the winner is deducted by her/his bid amount for that item.

Bid section: This section also starts with a line containing a single number $n < 100$, the number of bids. Each next n lines contains item id, bidder id, bid amount and bid time all separated with spaces. This means given bidder proposes the given bid amount at the given bid time for the given item. Their formats are as described above. Recall that a bidder can buy a item (win the auction for the item) if he propose a bid not later than the auction end time of the item and his bid amount is not less than the minimum price of the item and moreover his bid is highest among all valid bids for the item. Note that at the auction end time of the item, the winner is specified and the account-balance of the winner is updated.

You are guaranteed that no two items have the same auction end time and no two bids have the same bid time and bid amount. Note that bidder ids and item ids are unique but a bidder id can be equal to an item id and you may assume all ids in the input are valid.

Output (Standard Output)

Output one line for each item being auctioned, in order of their auction end time, printing "Item <item id> Bidder <bidder id> Price <winning bid amount> ". Note there should be a space between any two words in the output. If there is not a winning bid for an item, print "Item <item id> is not sold". Price in the output must be printed with exactly 2 digits after the decimal point.

Sample Input and Output

Standard Input	Standard Output
2 1 10.00 04:27:31 5 31.00 19:25:44 2 13 41.33 95 77.77 3 1 13 60.00 02:26:32 5 13 41.21 04:45:21 5 95 51.00 08:43:25	Item 1 is not sold Item 5 Bidder 95 Price 51.00

Problem G: Electing SSC Chair

Student Scientific Committee (SSC) has several members. In their first meeting, the SSC Chair is elected by the following rule. Every member votes and gives an ordered list of other members (including him/herself) who are candidate to be SSC chair. An election subcommittee (ES) will announce the winner (if there exists any) following this procedure which may be repeated in several rounds.
In the first round, ES will consider only the first choices in each member's vote list. If a candidate gets more than half of the votes, he/she wins the election. Otherwise, the candidate with the lowest vote counts (or candidates, in the case of a tie for fewest votes) is eliminated from all vote lists and the procedure is repeated. This is continued until a winner is found, or ES realizes there is no winner. Note in each round empty vote lists are non-viable and if a candidate is the first choice in more than half of viable vote list, the candidate is winner.

Input (Standard Input)

There are multiple test cases in the input. For each test case, first the number of candidates and the number of voter (all SSC members) are given respectively in one line separated with spaces (both values are at most 1000). Next vote lists are coming; a line for the vote list of each member. For each vote list, the names of the candidates are listed in the order of preference. A candidate may not appear in a voted list. A candidate name is an alphanumeric (case-sensitive) string with no space in middle. No candidate is repeated on a single vote. The input terminate with "0 0".

Output (Standard Output)

For each test case, output a single line containing the name of the winner as it is in the input. If no one wins, print "There is no winner".

Sample Input and Output

Standard Input	Standard Output
3 8 Saeed Ali Saeed Ali Saeed Ali Mohammad Ali Mohammad Ali Mohammad Ali Ali Saeed Ali Saeed 2 2 Ali Reza Reza Ali 0 0	Saeed There is no winner

Problem H: Mastermind

Mastermind is a code-breaking game for two players. One player is the *code-maker*, the other one is the *code-breaker*. The code-maker secretly chooses a *code*: an array of N integers $c_1, c_2, \dots, c_N$, each with a value in the range $[1, k]$ (Some numbers of $[1, k]$ may not appear in the code, and some may appear more than once). N and k are set at the beginning of the game. The code-breaker tries to find the code through a number of guesses. Each guess itself is also similar to the code; an array of N integers $g_1, g_2, \dots, g_N$ in the range $[1, k]$. After each guess, the code-maker provides the code-breaker with a hint showing the extent to which the guess matched the code. The hint consists of two numbers: the *black points*, and the *white points*. The black points is the number of correctly guessed slots, i.e. the number of indices i ($1 \le i \le N$) such that $g_i = c_i$. The white points shows the number of correctly guessed integers which appear in the code but not at the place they were in the guessed array. Each integer slot of the code (c_i) must be considered at most once during the calculation of the two hint points together, with a higher priority for the black points. For example if the code is "12234" and the guess is "22542", then the black and white points will be 1 and 2 respectively.
You have just joined a friend of yours while she was playing Mastermind as the code-breaker. After a number of guesses and receiving the hints, she is asking if you could help her find all the possible solutions according to the current state of the game. You must write a program to find any possible code which is consistent with all the guesses and their respective hints.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing three integers N, k, and g the number of guesses made up-to-now ($1 \le N \le 20$, $1 \le k \le 9$, $0 \le G \le 100$, and $k^N \le 10^5$). Then, g lines follow, showing the guesses and their respective hints. Each line starts with a string of N digits (in the range $[1, k]$) as the guess, followed by two space-separated integers as the black points and the white points for that guess. The input terminates with a line of the form "`0 0 0`" which should not be processed as a test case.

Output (Standard Output)

For each test case, write all the possible solutions of the game in the lexicographic order. Each solution must be printed as a string of N digits (in the range $[1, k]$) in a single line. On the last line of output for each test case, write the total number of possible solutions in the format "`Total: x`".

Sample Input and Output

Standard Input	Standard Output
4 4 1 1234 2 2 4 4 2 1234 2 2 4321 2 2 3 2 1 111 2 0 3 3 1 111 2 1 0 0 0	1243 1324 1432 2134 3214 4231 Total: 6 1324 4231 Total: 2 112 121 211 Total: 3 Total: 0