Problem A : Linear Equation (blue balloon)

You have been asked to write a program that can solve a simple linear equation.

Standard Input

The first line of input contains a single integer **P**, ($1 \leq \mathbf{P} \leq 1000$), which is the number of data sets that follow. Each data set consists of a single line containing one simple linear equation. All equations are strings of less than 200 characters. Each equation will be in the form of **a*x*,** followed by a single space, followed by a sign "+", followed by **b**, followed by a single space ,followed by a sign "=", followed by a single space, followed by **c**.

$$ax + b = c$$

where ***x*** is the variable (real number) and **a**, **b**, **c** are positive integers.

Standard Output

For each data set, generate two lines of output. The first line will contain "Equation **n**" where **n** is the number of the data set. The second line will contain the following answer:

- If the equation has no solution, print "No solution.".
- If the equation has infinitely many solutions, print "More than one solution.".
- If the equation has exactly one solution, print "x = *solution*" where *solution* is replaced by the appropriate real number (printed to six decimals).

Print a blank line after each data set case.

Sample Input	Sample Output
5 2x + 3 = 4 124x + 20 = 160 123456x + 7 = 2000 0x + 2 = 3 0x + 2 = 2	Equation 1 x = 0.500000 Equation 2 x = 1.129032 Equation 3 x = 0.016143

	Equation 4 No solution. Equation 5 More than one solution.

Problem B: Vote

(red balloon)

Benin is organizing a presidential election the 06th march 2016. Several candidates submitted their applications for this presidential election. The CENA called "Commission Nationale Electorale Autonome" is responsible for managing the election.

According to Article 15 of the beninese electoral code, the CENA is responsible for the preparation, the organization, the supervision of voting and for the centralization of the results.

As a great programmer, you are asked to help the CENA to centralize the results of the presidential election.

Standard Input

The first line of input contains a single integer **P**, ($1 \leq \mathbf{P} \leq 1000$), which is the number of data sets that follow. Each data set consists of a line containing the number *n* of the candidates ($1 \leq \boldsymbol{n} \leq 100$), a space and the number *m* of results to centralize ($1 \leq \boldsymbol{m} \leq 1000$) and followed by *n* lines and *m* lines. The *n* lines contain the names of candidates, one per line. The *m* lines contain each a name **X** of one candidate, a space, the result **R** of the candidate and the center **C** of vote. **X** and **C** are strings that will contain at most 1000 characters. **R** is a positive integer.

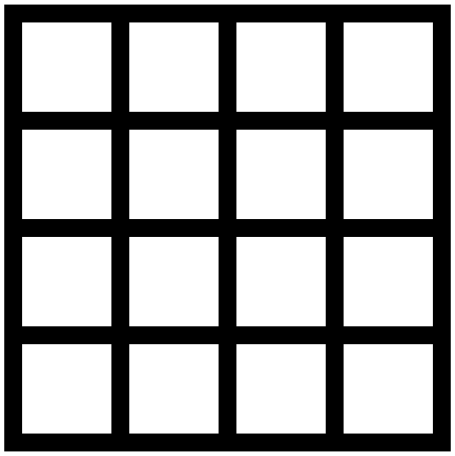
Standard Output

For each data set, if there is only one winner, generate one line of output with the text "VOTE *i*: THE WINNER IS " followed by the name of the winner, followed by space and followed by the total result of the winner. If not, generate the following output: "VOTE *i*: THERE IS A DILEMMA". *i* is the number of the data set.

Sample Input	Sample Output
2 3 4 Bignon Akwaba Sessi Bignon 1000 Gbegamey Sessi 1000 Yenawa Akwaba 5 Vodje Akwaba 996 Yenawa 2 3 Sena Sedjro Sedjro 6003 Malanville Sena 6000 Kpankpan Sena 3 Godomey	VOTE 1: THE WINNER IS Akwaba 1001 VOTE 2: THERE IS A DILEMMA

Problem C : Square (yellow balloon)

Do you know a game called “La cave aux énigmes”? One of its questions is to find the number of squares contained in a grid square of length *l*. A grid square of length 4 will look like this:



The total number of squares that can be seen in this image is 30. Your task is to find the total number of squares which can be seen in an image of a grid square of length *l*.

Standard Input

The input will begin with a single integer **P** on the first line, indicating the number of cases that will follow.

The remaining lines of the input will consist of one integer ***l*** per line, which is the grid square length. All integers will be less than 1,000,000 and greater than 0.

You should process all integers and for each integer *l*, determine the total number of squares which can be seen in an image of a grid square of length *l*.

You can assume that no operation overflows a 32-bit integer.

Standard Output

For each integer *l*, you should output the total number of squares which can be seen in an image of a grid square of length *l*, with one line of output for each line of input.

Sample Input	Sample Output
4 1 2 3 4	1 5 14 30

Problem D : Prison Break (green balloon)

A prison has been built as a labyrinth.

The labyrinth is composed of huts labelled + or *. If you are in hut +, you can only move to another hut + near your hut. One hut is considered near one another if the two huts have a side in common. If you are in hut *, you can only move to another hut * near your hut.

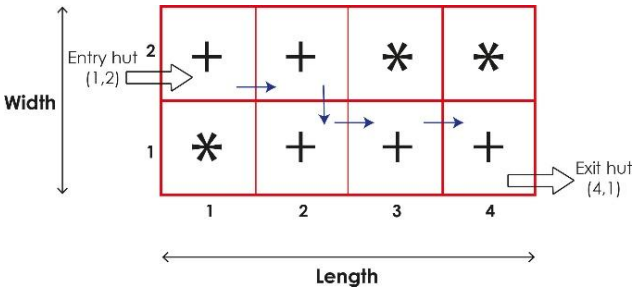
The labyrinth can be seen as a rectangle of huts of width **W** and length **L**. **W** and **L** are integers.

A hut is identified by its position on the horizontal side and by its position on the vertical side of the labyrinth.

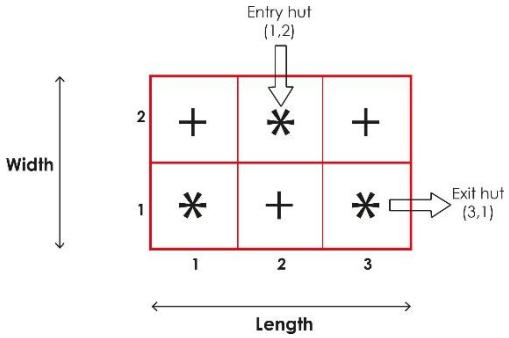
In this prison, all huts are “entry huts” but there is only one “exit hut”.

Given the labyrinth, the hut of the prisoner (called entry hut) and the exit hut, your task is to determine if the prisoner can escape.

In the example of Prison A (labyrinth of length 4 and width 2), the prisoner can escape. In the second example (labyrinth of length 3 and width 2), the prisoner cannot escape.



Prison A



Prison B

Standard Input

The input will begin with a single integer **P** on the first line, indicating the number of cases that will follow.

Each case begins with a single line made of 6 natural numbers with the following format:

L W A B C D where :

- **L** is the length of the labyrinth and **W** is the width of the labyrinth
- **A** is the length of the entry hut and **B** is the width of the entry hut of the prisoner
- **C** is the length of the exit hut and **D** is the width of the exit hut

followed by **W** lines containing **L** characters. Each character will be + or *.

Standard Output

For each prison, print YES if the prisoner can escape and NO if not.

Sample Input	Sample Output
2 4 2 1 2 4 1 ++** *+++ 3 2 2 2 3 1 +*+ *+*	YES NO

Problem E : X X glued (white balloon)

Do you know the game "X X glued" practiced on the registration plates of Benin cars by children?

A beninese car registration number is written in the form of VY ABCD RB or Y ABCD RB. For example, AC 2554 RB and X 6006 RB are beninese car registration numbers. The game "X X glued" is to quickly verify that there is in the number two identical consecutive digits or not. The winner is the first child that pronounces "X X collés" and shows the car with his finger.

Standard Input

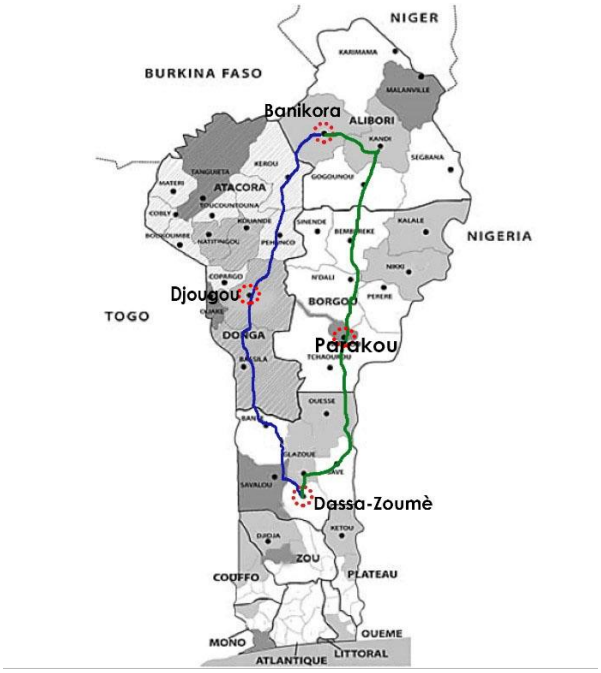
The input contains several lines and ended with the # character on a single line. Other lines each contain a single registration number as VY ABCD RB or Y ABCD RB.

Standard Output

For each registration number, your program should display on one line, all the " X X glued" in the format shown in the example. If there is no "X X glued" in the number, the program should display nothing.

Sample Input	Sample Output
X 6006 RB AC 2233 RB T 2000 RB F 2345 RB AB 4444 RB #	0 0 glued 2 2 glued and 3 3 glued 0 0 glued 4 4 glued

Problem F : The fastest road to banikoara (Orange balloon)



Codjo: I am ready for the trip to Banikoara

Bossi: Let us go through Parakou

Assiba: No, the fastest road to go to Banikoara is through Djougou

You are responsible to write a program which, given a list of towns and distances separating these towns, gives the shortest distance to travel from a town A to a town B.

Standard Input

The first line of input contains a single integer **P**, ($1 \leq \mathbf{P} \leq 1000$), which is the number of data sets that follow. Each data set begins with a line containing the number **N** of the remaining lines in the dataset ($1 \leq \mathbf{N} \leq 500$), followed by a space, followed by the name of a departure town, followed by a space, followed by the name of a destination town. Each of these N lines contains a name of a departure town, followed by a space, followed by a name of a destination town, followed by the distance (in kilometers) between the two towns. The distance will be an integer and the name of town will be a string formed with characters [a-z] [A-Z] and with the sign "-".

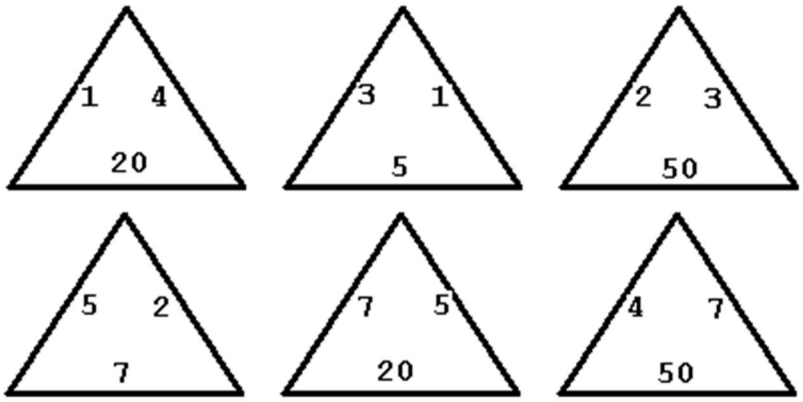
Standard Output

For each data set, you must generate a single output line containing the name of a town A and a space, followed by the name of town B, followed by a space, followed by the shortest distance to travel between towns A and B.

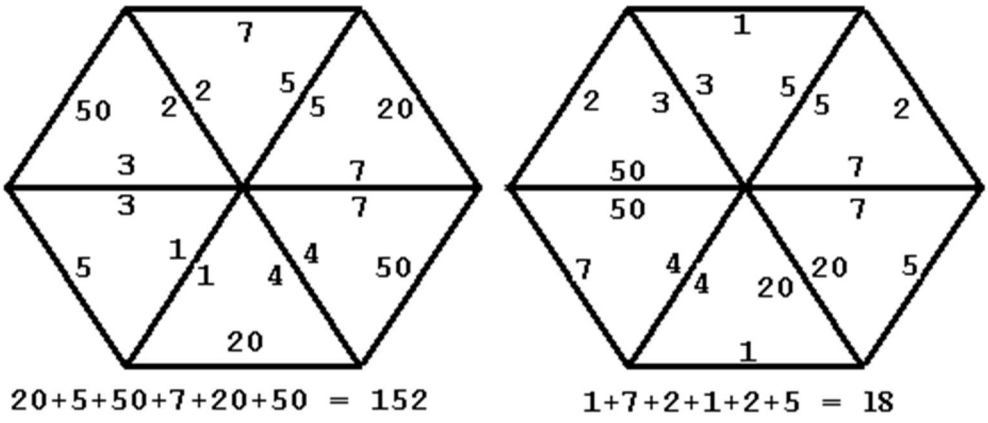
Sample Input	Sample Output
1 4 Dassa-Zoume Banikoara	Dassa-Zoume Banikoara 481

Dassa-Zoume Djougou 270 Banikoara Djougou 211 Parakou Banikoara 284 Parakou Dassa-Zoume 225	

Problem G : The Triangle Game (Purple balloon)



In the triangle game you start off with six triangles numbered on each edge, as in the example above. You can slide and rotate the triangles so they form a hexagon, but the hexagon is only legal if edges common to two triangles have the same number on them. You may not flip any triangle over. Two legal hexagons formed from the six triangles are illustrated below.



The score for a legal hexagon is the sum of the numbers on the outside six edges.

Your problem is to find the highest score that can be achieved with any six particular triangles.

Standard Input

The input file contain one or more data sets. Each data set is a sequence of six lines with three integers from 1 to 100 separated by blanks on each line. Each line contains the numbers on the triangles in clockwise order. Data sets are separated by a line containing only an asterisk. The last data set is followed by a line containing only a dollar sign.

Standard Output

For each input data set, the output is a line containing only the word "none" if there are no legal hexagons or the highest score if there is a legal hexagon.

Sample Input	Sample Output
1 4 20 3 1 5 50 2 3 5 2 7 7 5 20 4 7 50 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 60 6 10 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 10 6 60 $	152 21 none

Source: MCPC 2000

Problem H : RIPOFF
(Black balloon)

Business has been slow at Gleamin' Lemon Used Auto Sales. In an effort to bring in new customers, management has created the Rebate Incentive Program Of Fabulous Fun (or RIPOFF). This is a simple game which allows customers to try and win a rebate on an automobile purchase. The RIPOFF game is a board game where each square is labeled with a rebate amount. The customer advances through the board by spinning a spinner. Each square he lands on adds to his total rebate amount. When he reaches the end of the board he is rewarded with the total rebate amount.

Of course, given the company involved, it should come as no surprise that there are a couple of catches written in the fine print. The first is that there is a limit to the number of turns the customer has to finish the game; if he doesn't reach the end within the allotted number of turns then he loses his rebate. The second is that some of the squares actually have a negative amount which subtract from the rebate instead of adding to it. A particularly unlucky customer might even come out of the game with a negative rebate.

Even with these catches, the management of Gleamin' Lemon is concerned that someone might win a particularly large rebate—something they would like to avoid at all costs. Your job is to take a particular configuration for the RIPOFF game and decide the maximum rebate a customer could possibly obtain.

Consider, for example, the game board below. Assume we have 5 turns to finish the game, and each turn we can move between 1 and 4 spaces depending on what we spin. Notice that we must start just before the board begins, so spinning a 1 causes us to land on the first square. Also notice we must end by landing past the end of the last square. It does not have to be exact; any number that gets us off of the board will work.

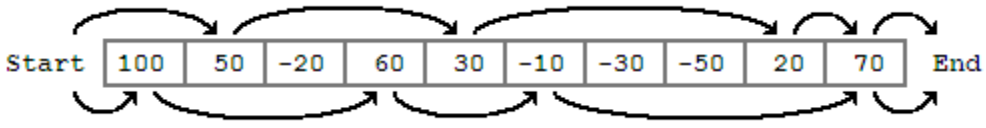


Figure 1

The illustration shows two different possible ways the game might go. Following the arrows on the top, if we spin a 2, 3, 4, 1, and 1 respectively, we will win a total rebate of 50 + 30 + 20 + 70 = $170. However, the best possible rebate we could win would be $220. We would win this amount if we spun a 1, 3, 2, 4, and 1 respectively, as shown by the lower path. Notice that we did not land on every square with a positive number; if we had we wouldn't have been able to make it to the end of the board before the 5 turns was up.

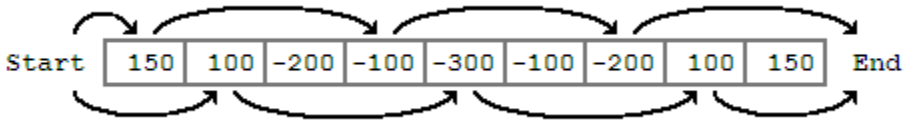


Figure 2

The illustration in Figure 2 shows a game where we have 4 turns to finish the game, and can move up to 3 spaces each turn. Again, two different paths are shown, the one on top earning a rebate of -$150, and the one on bottom earning a rebate of -$100. In fact, -$100 is the highest possible rebate we could earn for this game (a fact that would no doubt please the management of Gleamin' Lemon). Of course, there also might be a sequence of moves in which we do not reach the end before the turn limit—e.g. spinning a 1 every time. Although not finishing would actually be preferable to finishing with a negative rebate, in this problem we are only going to consider sequences of moves which allow us to reach the end before the turn limit.

Standard Input

The input consists of one to twenty data sets, followed by a line containing only 0. The first line of a data set contains three space separated integers **N S T**, where

- **N** is the total number of squares on the board, $2 \le N \le 200$.
- **S** is the maximum number of spaces you may advance in each turn, $2 \le S \le 10$.
- **T** is the maximum number of turns allowed, where $N + 1 \le ST$ and $T \le N + 1$.

The data set ends with one or more lines containing a total of N integers, the numbers on the board. Each number has magnitude less than 10000.

Standard Output

The output for each data set is one line containing only the maximum possible rebate that can be earned by completing the game.

To complete the game you must advance a total of $N + 1$ spaces in at most T turns, each turn advancing from 1 to S spaces inclusive. It will always be possible to complete a game. However, there may be a very large number of different turn sequences that will finish, so you will need to be careful in choosing your algorithm.

The sample input data corresponds to the games in the Figures.

Sample Input	Sample Output
10 4 5 100 50 -20 60 30 -10 -30 -50 20 70 9 3 4 150 100 -200 -100 -300 -100 -200 100 150 0	220 -100

Source: MCPC 2009