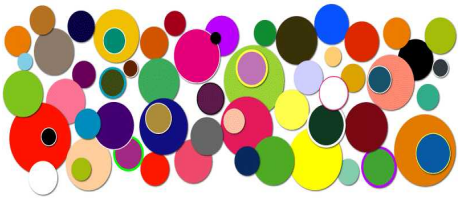
Problem A : Colors



A color reduction is a mapping from a set of discrete colors to a smaller one. The solution to this problem requires that you perform just such a mapping in a standard twenty-four bit RGB color space. The input consists of a target set of sixteen RGB color values, and a collection of arbitrary RGB colors to be mapped to their closest color in the target set. For our purposes, an RGB color is defined as an ordered triple (*R*,*G*,*B*) where each value of the triple is an integer from 0 to 255. The distance between two colors is defined as the Euclidean distance between two threedimensional points. That is, given two colors (*R*1,*G*1,*B*1) and (*R*2,*G*2,*B*2), their distance *D* is given by the equation

$$D = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2} \ .$$

Input

The input is a list of RGB colors, one color per line, specified as three integers from 0 to 255 delimited by a single space. The first sixteen colors form the target set of colors to which the remaining colors will be mapped. The input is terminated by a line containing three –1 values.

Output

For each color to be mapped, output the color and its nearest color from the target set.

Sample Input	Sample Output
0 0 0 255 255 255 0 0 1 1 1 1 128 0 0 0 128 0 128 128 0 0 0 128 126 168 9 35 86 34 133 41 193 128 0 128 0 128 128 128 128 128	(0,0,0) maps to (0,0,0) (255,255,255) maps to (255,255,255) (253,254,255) maps to (255,255,255) (77,79,134) maps to (128,128,128) (81,218,0) maps to (126,168,9)

255 0 0 0 1 0 0 0 0 255 255 255 253 254 255 77 79 134 81 218 0 -1 –1 -1	

Problem B: Jailer

One prison contains a long hall of ***n*** cells, each right next to each other. Each cell has a prisoner in it, and each cell is locked. One night, the jailer gets bored and decides to play a game. For round 1 of the game, he takes a drink of whiskey, and then runs down the hall unlocking each cell. For round 2, he takes a drink of whiskey, and then runs own the hall locking every other cell (cells 2, 4, 6, …). For round 3, he takes a drink of whiskey, and then runs down the hall. He visits every third cell (cells 3, 6, 9, …). If the cell is locked, he unlocks it; if it is unlocked, he locks it. He repeats this for ***n*** rounds, takes a final drink, and passes out. Some number of prisoners, possibly zero, realizes that their cells are unlocked and the jailer is incapacitated. They immediately escape.
Given the number of cells, determine how many prisoners escape jail.

Input

The first line of input contains a single positive integer. This is the number of lines that follow. Each of the following lines contains a single integer between **5** and **100**, inclusive, which is the number of cells ***n***.

Output

For each line, you must print out the number of prisoners that escape when the prison has ***n*** cells.

Sample Input	Sample Output
2 5 100	2 10

Problem C: K in a row

Poxos and Petros do not listen to their teacher at school. Instead they secretly play a game called K_In_A_Row. One day in the evening they started to argue who had won more games that day. They collected all the papers they had used for playing the game and they started to count how many times each of them had won. But it was very tedious and they were sleepy. Help them to count how many games each of them had won.

K_In_A_Row is played in a square grid with **M** times **N** squares. Two players alternate in their moves. A player chooses an empty square and fills in his/her sign (Poxos uses cross 'x' and Petros uses circle 'o'). The game is won by the player who first places at least **K** his/her own signs in a row (either horizontally, vertically or in one of the two diagonal directions). The game stops immediately after one of the players completes **K** of his/her signs in a row; thus it may never happen that both players have completed **K** of their signs in a row. If no player creates such a row, nobody wins.

Input

The first line contains the number of games **L**. It is followed by **L** blocks, each describing one game. Each block starts with a line containing 3 numbers **M**, **N** and **K**. Numbers **M** and **N** give the size of the grid (0<**M,N<50**) and **K** is the length of the required row. The following **N** lines each containing **M** characters describe the situation after the end of the game. Character '.' denotes an empty field, characters 'x' and 'o' denote fields marked by Poxos and Petros respectively. You may assume that the input is correct.

Output

The output consists of two numbers separated by a colon ':'. The first number denotes the number of the games won by Poxos, the second one gives the number of the games won by Petros.

Sample Input	Sample Output
2 3 3 3 .x. .xo oox 4 7 4 .... ..x. ooox oxx. oox. o.ox ..xx	0:1

Problem D: Gossipers

Yerevan is a big town in the middle of Armenia; what makes it so famous is the number of gossipers who live there. Every morning, each gossiper finds out a new gossip, a gossip so unique that nobody else in the town knows it. The gossipers talk, gossip and exchange rumors all day long. What happens when two gossipers meet? Of course, they exchange all the gossips they have heard so far. Your task is to determine whether every gossiper will know all the gossips by the end of the day.

Input

The input consists of multiple test cases separated by blank lines. On the first line of every test case there are two positive integers **N** and **M**, where **N** is the number of gossipers and **M** is the number of meetings. 2 <= *N* <= 50 2 <= M <= 300.. On the next **N** lines there are the names of the gossipers. The name of each gossiper is a single word consisting of lower- and uppercase letters. The following **M** lines describe the meetings in the order they happened. Each meeting is described by two distinct names of the gossipers separated by a single space. The values **M=N=0** indicate the end of the input file.

Output

The output should contain for each test case one line containing a single word "YES" if every gossiper knows all the gossips, or "NO", otherwise.

Sample Input	Sample Output
3 3 Alice Bob Cindy Alice Bob Bob Cindy Cindy Alice 4 4 Kirk Lucy Mike Nancy Kirk Lucy Lucy Mike Mike Nancy Nancy Lucy 0 0	YES NO

Problem E: Combination Lock

A combination lock consists of a circular dial, which can be turned (clockwise or counterclockwise) and is embedded into the "fixed" part of the lock. The dial has N evenly spaced "ticks". The ticks are numbered from 0 to N-1, increasing in the clockwise direction. The fixed part of the lock has a "mark" which always
"points to" a particular tick on the dial. Of course, the mark points to different ticks as the dial is turned.
The lock comes with three code numbers T1, T2, T3. These are non-negative integers and each of them is less than N. No two of the three are the same.

The lock is opened in three stages of operations:

1. Turn the dial clockwise exactly two full revolutions, and continue to turn it clockwise until the mark points to tick T1.
2. Turn the dial one full revolution counterclockwise and continue to turn it counterclockwise until the mark points to tick T2.
3. Turn the dial clockwise until the mark points to tick T3. The lock should now open.

You must find the maximum possible number of ticks the dial must be turned in order to open the lock. The number of ticks turned is defined to be the sum of the ticks turned in the three stages outlined above, and is always positive regardless of direction.

Input

The input consists of a number of test cases, one test case per line. Each line of the input file contains four integers: *N*, *T1*, *T2*, *T3*, in this order, separated by blank spaces. The integer *N* is a multiple of 5, 25 <= *N* <= 100. The numbers *T1*, *T2* and *T3* satisfy the constraints stated under the description above. The input will be terminated by a line with four blank-separated 0's.

Output

For each test case, print the maximum possible number of ticks the dial must be turned in order to open the lock. Print each on its own line. There should be no blank lines between outputs.

Sample Input	Sample Output
80 20 40 50 80 10 79 12 0 0 0 0	409 455

Problem F: Jack's Lotto Tickets

Jack likes to play the lotto. Whenever he does, he buys lots of tickets. Each ticket has 6 unique numbers in the range from **1** to **49**, inclusive. Jack likes to "Cover all his bases." By that, he means that he likes for each set of lottery tickets to contain every number from **1** to **49**, at least once, on some ticket. Write a program to help Jack see if his tickets "Cover all the bases."

Input

The input consists of a number of test cases. Each case starts with an integer ***N*** (**1** <= ***N*** <= **100**), indicating the number of tickets Jack has purchased. On the next ***N*** lines are the tickets, one per line. Each ticket will have exactly 6 integers, and all of them will be in the range from **1** to **49** inclusive. No ticket will have duplicate numbers, but the numbers on a ticket may appear in any order. The input ends with a line containing only a **0**.

Output

Print a list of responses for the input sets, one per line. Print the word **Yes** if every number from **1** to **49** inclusive appears in some lottery ticket in the set, and **No** otherwise. Print these words exactly as they are shown. Do not print any blank lines between outputs.

Sample Input	Sample Output
1 1 2 3 4 5 6 9 1 2 3 4 5 6 10 9 8 7 12 11 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 19 34 27 25 13 0	No Yes

Problem G: Check

You must check if $(m-1)! + 1$ divides m or not, where m is integer number !
N! = 2*3*..*(N-1)*N

For example when m = 5 then (5-1)! + 1 = 25 and 25 divides 5.

Input

Input contains one integer number m ($1 < m < 10^9$).

Output

Output "YES" if (m-1)! + 1 divides m, and "NO" otherwise

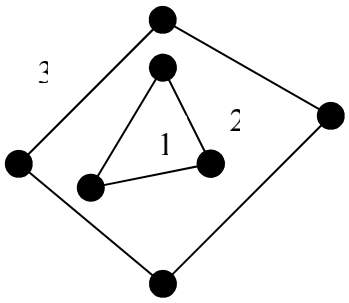
Sample Input	Sample Output
5	YES
6	NO

Author: Eduard Piliposyan

Problem H: How Many Parts ?

There are N distinct points in the plain, and some of them are connected using straight line interval. It is known that no 2 intervals intersect (except in the given points).

For example.



In how many parts does it divide the plain. In the given example answer is 3, and three parts are shown in the picture by numbers.

Input

The first line of the input contains two integer numbers N and M ($1 \le N \le 50$) ($1 \le M \le 500$) number of points and number of line intervals. Each of the next N lines contains two integer numbers X_i, Y_i coordinate of the i-th point ($0 \le X_i, Y_i \le 50000$). Each

of the next M lines contain 2 integer numbers u , v from range 1..N. It means that point X_u,Y_u is connected with point X_v,Y_v with straight line interval.

Output

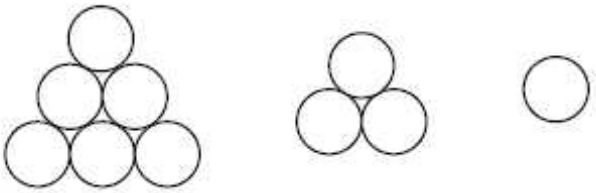
Output number of parts of the plain.

Sample Input	Sample Output
7 7 **2 4** **6 7** **10 3** **5 1** **4 3** **6 5** **7 3** **1 2** **2 3** **3 4** **4 1** **5 6** **6 7** **7 8**	**3**

Author: Eduard Piliposyan

Problem I: Cannonball Pyramids

If you visit historical battlegrounds from periods when cannon were used, you may see cannonballs (metal spheres) stacked in various ways. In this problem we're interested in the number of cannonballs in such pyramids with triangular bases — that is, with cannonballs arranged in an equilateral triangle on the base, then similar equilateral triangles of cannonballs stacked on top of that, until a single cannonball is placed on the top. This three-sided pyramid is, in fact, one of the Platonic solids, a tetrahedron. For example, suppose we begin with a base containing 6 cannonballs arranged in an equilateral triangle as shown in the figure on the left; this equilateral triangle has 3 cannonballs on each edge. The next layer will contain 3 cannonballs, 2 on each edge — as shown in the middle. The final layer always contains just one cannonball. The total number of cannonballs in this pyramid is thus 6 + 3 + 1 = 10. In general, each layer of such a pyramid will have one fewer cannonballs on each edge of the triangle than the next lower layer.



The number of cannonballs in a layer is just the sum of the integers from 1 to N, where N is the number of cannonballs on one side of the layer. Given the number of cannonballs on each side of the base, compute the total number of cannonballs in the entire tetrahedral pyramid.

Input

The first line of the input contains a single integer N giving the number of cases posed, for a maximum of 100 cases. Following that are exactly N lines, each with a single integer giving the number of cannonballs on each side of the base for a tetrahedron of cannonballs, a number less than 1000.

Output

For each input case, display the case number (1, 2, …), a colon and a blank, the number of cannonballs on each side of the base, one blank, and finally the total number of cannonballs in the tetrahedron.

Sample Input	Sample Output
2 3 5	1: 3 10 2: 5 35

Problem J: Even Parity

A bit string has an *odd parity* if the number of 1's is odd. A bit string has *even parity* if the number of 1's is even. Zero is considered to be an even number, so a bit string with no 1's has even parity. Note that the number of 0's does not affect the parity of a bit string.

Input

The input consists of one or more strings, each on a line by itself, followed by a line containing only "#" that signals the end of the input. Each string contains 1–31 bits followed by either a lowercase letter 'e' or a lowercase letter 'o'.

Output

Each line of output must look just like the corresponding line of input, except that the letter at the end is replaced by the correct bit so that the entire bit string has even parity (if the letter was 'e') or odd parity (if the letter was 'o').

Sample Input	Sample Output
101e 010010o 1e 000e 110100101o #	1010 0100101 11 0000 1101001010