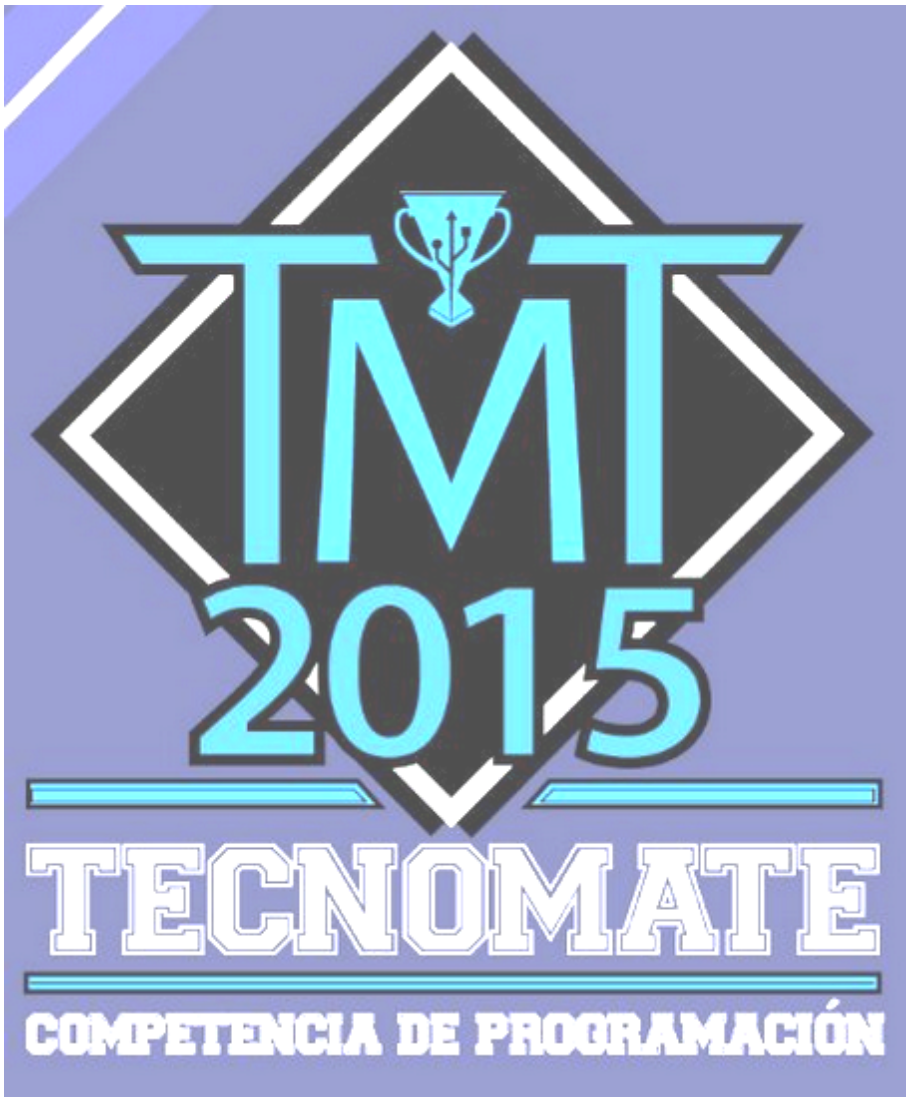
Departamento de Ingeniería en Sistemas de Información.
Facultad Regional Santa Fe
Universidad Tecnológica Nacional



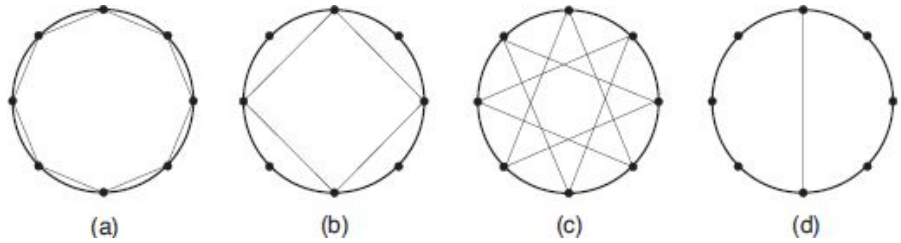
Nivel “Libres”

18 de Setiembre

Problem 1: Star (STARSBC)

Fernando won a compass for his birthday, and now his favorite hobby is drawing stars: first, he marks **N** points on a circumference, dividing it into **N** equal arcs; then, he connects each point to the *k-th* next point, until returning to the first point.

Depending on the value of *k*, Fernando may or may not reach all points marked on the circumference; when that happens, the star is called complete. For example, when **N** = 8, the possible stars are shown in the figure below. Stars (a) and (c) are complete, while stars (b) and (d) are not.



Depending on the value of **N**, it may be possible to draw many different stars; Fernando asked you to write a program that, given **N**, determines the number of complete stars he can draw.

Input

The input contains several test cases. Each test case contains a single line, containing a single integer N ($3 \leq N < 2^{31}$), indicating the number of arcs in which the circumference was divided.

Output

For each test case, your program must print a single line containing a single integer, indicating the number of complete stars that can be drawn.

Example

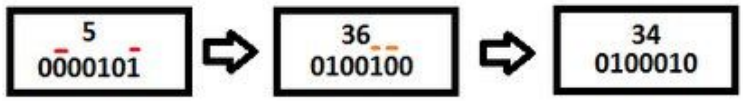
Input	Output
3 4 5 18 36 360 2147483647	1 1 2 3 6 48 1073741823

Problem 2: Bit Shuffling (BITSHUFFLING)

To find out if his students really understood the lecture about binary representation of integer numbers, Teacher Marcelo came up with the following problem:

"Given an integer number and a sequence of bit permutations for its binary representation, find 3 numbers: the resulting number after all permutations, the maximum and minimum values found during the permutations".

Teacher Marcelo promised one extra point to whoever solved the problem first. Since he never did such thing before (giving extra points), you rushed to solve the problem as quick as you could, fearing the Professor could change his mind.



Input

The first line of a test case contains the integers **N (0 ≤ N ≤ 2^{32} - 1)** and **K (1 ≤ K ≤ 100)**, representing the starting number and the number of permutations, respectively. Each of the following **K** lines will contain two space separated integers **A** and **B (0 ≤ A, B ≤ 31)**, indicating that bits **A** and **B** must be swapped. Input ends when N = K = 0.

Output

For each test case print 3 integers separated by space: **RES MAX MIN**, where **RES** is the number **N** after all permutations, **MIN** and **MAX** are, respectively, the smallest and greatest intermediate value found in the permutation process. (**MAX** and **MIN** must consider the first and last value **N** assumed as well.)

Example

Input	Output
5 2 0 5 1 2 0 0	34 36 5

Problem 3: Focus (VARYINGFOCUS)

Daniel is taking a Computacional Vision course and decided to reproduce an interesting work that he saw at class: he took some photos of a same scene, varying just the focus, for later to combine them in just one photo, in which all the objects in scene are clear together. For that, he needs that each object seem clearly in at least one photo.

Daniel knows that for each object, there is a closed interval of focus plans in which that object stays clearly visible. For example, in the picture below, (i), (ii) and (iii) are tree photos of the same scene, each one taken with a different focus; (iv) is the combined image generated by Daniel.

As the memory card of Daniel's camera has a small capacity, he asked for your help. Given the intervals of focus of all the objects in the scene that he pretends to photograph, determine the minimum number of photos that he should take so that each object stays clear in at least one of the pictures.

Input

The input consists in several case tests. The first line of each test case contains one integer **N** (1 ≤ **N** ≤ 106) that indicates the number of objects in the scene. Each one of the **N** next lines contains two integers, **A** and **B** (1 ≤ **A** ≤ **B** ≤ 109), that indicates the extremes of the focus interval of each object.

Output

For each test case, you should print one line with a integer that indicates the minimum number of photos that Daniel have to take.

Examples:

Input	Output
3 1 3 2 5 4 6 5 1 2 5 6	2 3

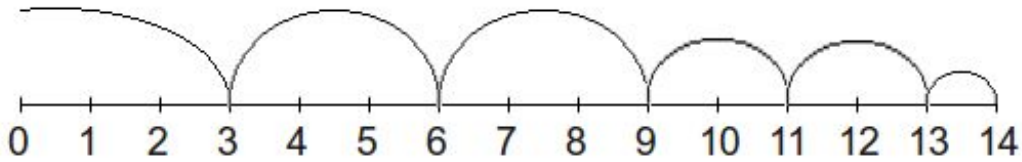
3 4 5 6 1 2	

Problem 4: Throwing Balls (THROWINGBALLS)

Your friends invented a new competition: Throwing Balls. The goal is simple, just throw a ball in a way that it falls into a hole N meters ahead.

When the ball is thrown, let's say that at an integer speed V, it stays in the air for V meters and then touches the ground. It repeats this process V times. After it touches the ground V times, it changes its speed to V-1, and the previous process repeats, until the speed is equal to 0.

For example, if the ball is thrown at a speed equal to 3, it will touch the ground at the following points: 3, 6, 9, 11, 13, 14; as it can be seen in the picture below.



You can throw a ball at an integer speed less than or equal to V. Given the distance of the hole, say if it's possible for you to throw the ball and that it falls exactly into the hole.

Input

There will be several test cases. Each test case contains two integers, N and V ($1 \leq N \leq 1000$, $1 \leq V \leq 30$), representing the distance of the hole, and the maximum speed that you can throw the ball.

The last test case is indicated when N = V = 0, which should not be processed.

Output

For each teste case, print one line containing the word "possivel" (without the quotation marks), if it is possible to throw the ball at an integer speed less than or equal to V, in a way that it falls into the hole, or "impossivel" otherwise.

Examples:

Input	Output
14 3 13 3 12 3 5 3 30 4 0 0	possivel possivel impossivel possivel possivel

Problem 5: Contamination (CONTAMINATION)

The year is 2241, and the colonization of other planets is now a reality. You work in the control center of resources on the planet URI-942, mainly controlling the water supplies. The water is stored in underground tanks, protected from the high surface temperatures.

But, your fellow colleagues Ana and Márcio found some shortcomings in the walls of tanks, which can lead to contamination of the water stock. Your colleagues were able to identify the points where there may be flaws with the infiltration of contaminants. Knowing that the contaminants spread throughout the water tank affected, your task is to estimate the contamination of the water according to the maps provided by their peers.

The maps were discretized into cells, and cells may correspond to a region of rock, water (tank) or a contaminant. Because the cracks, a cell with contaminant contaminate adjacent cells (left, right, above and below) containing water, but contamination is barred by cells of rock.

Input

The input consists of several maps, and a description of each map starts with a line containing two integers N and M, corresponding to the number of rows and columns of the map. The following N lines describe the map, each line containing M characters, beyond the jump line. Possible characters are A, which is a cell containing water, X represents a cell with rock and T represents a cell with contaminant.

The entry ends when N = M = 0, the case should not be processed. In all maps, N and M are less than or equal to 50.

Output

For each map, print an estimation of future contamination. This estimation should match the original map (as seen at the entrance), but replacing the cells with water that have been contaminated by the character T. Leave a blank line after each map (including the last map).

Example:

Input	Output
6 7 XXAAXXX XXAAXAX XXXXAXX XAAAAAX TAAXAAA XXXXXXX 3 3	XXAAXXX XXAAXAX XXXXTXX XTTTTTX TTTXTTT XXXXXXX TTT

TTT XXX AAA 0 0	XXX AAA

Problem 6: Elegant Permuted Sum (ELPESUM)

You will be given **n** integers $\mathbf{A_1A_2A_3...A_n}$. Find a permutation of these **n** integers so that summation of the absolute differences between adjacent elements is maximized.

Suppose **n = 4** and the given integers are **4 2 1 5**. The permutation **2 5 1 4** yields the maximum summation. For this permutation **sum = abs(2-5) + abs(5-1) + abs(1-4) = 3+4+3 = 10**.

Of all the **24** permutations, you won't get any summation whose value exceeds **10**. We will call this value, **10**, the *elegant permuted sum*.

Input

The first line of input is an integer **T** (**T** < 100) that represents the number of test cases. Each case consists of a line that starts with **n** (1 < **n** < 51) followed by n non-negative integers separated by a single space. None of the elements of the given permutation will exceed 1000.

Output

For each case, output the case number followed by the *elegant permuted summation*.

Example:

Input	Output
3 4 4 2 1 5 4 1 1 1 1 2 10 1	Case 1: 10 Case 2: 0 Case 3: 9

Problem 7: Variations (VARIATIONSTM)

The internet is not as safe as it was in the past. One evidence of it is the increase in the number of hacker attacks. To worsen, when a hacker steals a password from a user in a specific site, he has access to all the other accounts of this user in other sites, because most of the users nowadays use the same password in all sites.

One of the proposed solutions to this problem is to use different passwords for each site, or even different variations of the same password. For example, to vary the password "potato", it's possible to use the password "pOtaTo", "P0tat0", "pot4TO", etc. In other words, for each character of the alphabet, it's possible to make a variation changing the case of the character (lower case or upper case). Moreover, to increase the total number of variations, for the characters A, E, I, O and S it's possible to use the numbers 4, 3, 1, 0 and 5, respectively.

Your friend needs to increase the number of variations of his password, and asked for your help. Given the password that he chose, find out the number of different variations that it's possible to build.

Input

The first line contains an integer **T**, indicating the number of test cases to follow.

Each test case contains a sequence of characters **S**, indicating your friend's password. For each password, there will be at least 1 and at most 16 characters, which can be any of the 26 letters of the alphabet, lower or upper case.

Output

For each test case print one line containing one integer, indicating the number of different variations that is possible to build with the given password, including itself.

Example:

Input	Output
4 bB ab Ee bAtatA	4 6 9 216

Problem 8: Compare Substring (CMPSSTR)

Find the longest common substring between the two informed Strings. The substring can be any part of the String, including the entire String. If there is no common substring, return 0. The search is *case sensitive* ('x' != 'X').

Input

The input contains several test cases. Each test case is composed by two lines that contains a string each. Both input Strings will contain between 1 and 50, inclusive, letters (a-z, A-Z), and/or spaces.

Output

The length of the longest common substring between the two Strings.

Example:

Input	Output
abcdef cdofhij TWO FOUR abracadabra open Hey This java is hot Java is a new paradigm	2 1 0 7