

1. All fractions written in octal (base 8) notation may be expressed exactly in decimal notation. For example, 0.75 in octal is 0.953125 ( $7/8 + 5/64$ ) in decimal. Specifically, a numeral requiring  $N$  octal digits to the right of the octal point may always be written as a decimal numeral with no more than  $3N$  digits to the right of the point. The reverse, of course, is not true.

Write a program to convert non-negative octal numerals less than 1 (with leading '0.') into equivalent decimal numerals. The input to your program will consist of zero or more octal numerals in the specified range separated by whitespace. Each numeral has the form  $0.d_1d_2\cdots d_k$ , where the  $d_i$  are octal digits (0..7). There is no limit on  $k$ .

The output will consist of a sequence of lines—one for each input—having the form

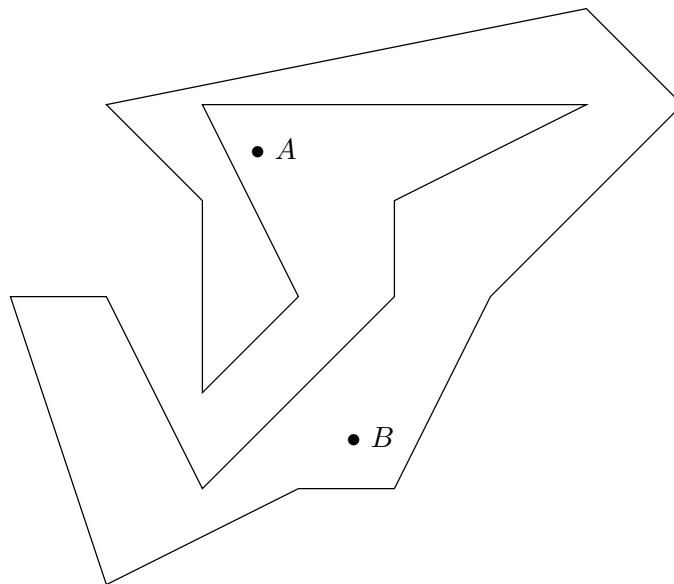
$$0.d_1\cdots d_k [8] = 0.D_1\cdots D_m [10]$$

where the left side echos the input and the right side is the base-10 equivalent. The right side must satisfy the condition  $D_m \neq 0$ .

Here is an example of the desired format.

Input	Output
0.75 0.0001	0.75 [8] = 0.953125 [10]
0.01234567	0.0001 [8] = 0.000244140625 [10]
	0.01234567 [8] = 0.020408093929290771484375 [10]

2. The infamous prisons on Haunted Simplex Island are all built out of unscalable walls in the shape of immensely elaborate simple (non-self-intersecting) polygons of enormous extent. The outsides of the walls look just like the insides, and the sadistic jailors often delight in placing the prisoner *outside* the jail wall (without telling him, of course), next to an immovable source of water. Not wanting to risk dying of thirst (it is very hot on the island), prisoners will generally stick close to the water rather than exploring to see what side of the wall they are on. If a prisoner *did* know that he was outside, he would probably risk searching for the exit. For example, prisoner *A* below is actually free, and *B* is not.



Given a description of the wall and a prisoner's position, determine whether the prisoner is actually free. The input to your program will consist of a pair of integer coordinates  $P_x$  and  $P_y$ , giving the prisoner's position, an integer,  $3 \leq N < 200$ , giving the number of walls, and  $N$  sets of coordinates,  $X_i$  and  $Y_i$ , of the corners where adjacent sections of the wall meet, where each coordinate may range from 0 to 100000. The corners are listed in clockwise order around the wall from an arbitrary starting point. There is an implicit wall section between the last corner listed and the first. You may assume the prisoner is not on the wall, that the wall segments do not intersect except at the corners (and that only two wall segments join there), and that (although it is really strong) the wall is infinitely thin.

Print out either "The prisoner is free" or "The prisoner is confined", as shown in the examples on the next page.

**Example 1.**

Input	Output
175 225	The prisoner is free
17	
100 250 350 300 400 250	
300 150 250 50 200 50	
100 0 50 150 100 150	
150 50 250 150 250 200	
350 250 150 250 200 150	
150 100 150 200	

**Example 2.**

Input	Output
225 75	The prisoner is confined
17	
100 250 350 300 400 250	
300 150 250 50 200 50	
100 0 50 150 100 150	
150 50	
250 150	
250 200	
350 250 150 250 200 150	
150 100 150 200	

3. [From the 1993 Internet Contest] Given a list of variable constraints of the form  $x < y$ , you are to write a program that prints all orderings of the variables that are consistent with the constraints. For example, given the constraints  $x < y$  and  $x < z$ , there are two orderings of  $x$ ,  $y$ , and  $z$  that are consistent with these constraints:  $(x, y, z)$  and  $(x, z, y)$ .

The input consists of a sequence of problems. Each problem consists of two lines of input, each in free form. The first is a list of distinct, single-character, lower-case variables (separated by whitespace). The second is also a list of an even number of variables from the first line in the same format, specifying the constraints. The list  $x_1 x_2 \cdots x_{2n+1}$  indicates the  $n$  constraints  $x_1 < x_2, \dots, x_{2n} < x_{2n+1}$  (the  $x_i$  here need not be distinct).

There will be at least two and most 20 variables in a problem. There will be at least one and no more than 50 constraints in a problem. There will be at least one and at most 300 orderings consistent with the constraints. Input is terminated by the end of file.

For each problem, print all the orderings in lexicographic (alphabetical) order, one per line in the format shown in the example.

**Example:**

Input	Output
x y z	Problem #1:
x y x z	xyz
a b f g	xzy
a b b f	Problem #2:
v w x y z	abfg
v y x v z v w v	abgf
	agbf
	gabf
	Problem #3:
	wxzvy
	wzxvy
	xwzvy
	xzwyv
	zwxvy
	zxwvy