CHOOSING ICE CREAM

You are standing in the supermarket in front of the freezers. You have a very tough task ahead of you: you have to choose what type of ice cream you want for after dinner that evening. After a while, you give up: they are all awesome! Instead, you take your (fair) k-sided die out of your pocket and you decide to let fate decide.

Of course, the number of ice cream choices, n, may not be precisely k, in which case you could not just throw the die once, rolling i, and take the ith ice cream choice. You therefore have to use some algorithm that involves zero or more die throws that results in an ice cream choice with every choice being exactly equally likely. Being a good computer scientist, you know about the accept-reject method, which would let you make such a fair choice.

At that point, you remember that you have a very important competition to attend that same afternoon. You absolutely cannot afford to be late for that competition. Because of this, you decide you cannot use the accept-reject method, as there may be no bound on the number of die throws needed to ensure a fair result, so you may end up standing there for a long time and miss the competition! Instead, you resolve to find an algorithm that is fair and uses as few dice choices as possible in the worst case.

Given n and k, can you determine the minimum number i such that there is a fair algorithm that uses at most i die throws per execution?

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n and k ($1 \le n, k \le 109$): the number of ice cream choices and the number of sides of your die, respectively.

Output

Per test case:

- one line with a single integer: the smallest number of throws after which you are guaranteed to be able to make a fair choice. If there is no such number, print “unbounded” instead.

Examples

№	stdin	stdout

1	3
	4 2
	2 4
	3 2
		2
		1
		unbounded