Vases
Input File: VasesIn.txt

Ryan's mom gave him a vase with a cherished inscription on it, which he often places on his window ledge. He is moving into a high rise apartment complex and wants to make sure that if the vase falls off the ledge, it won't break. So he plans to purchase several identical vases and drop them out of the window to determine the highest floor he can live on without the vase breaking should it fall off the window ledge. Time is short, so Ryan would like to minimize the number of drops he has to make.

Suppose that one vase was purchased and the building was 100 stories. If it was dropped out of the first floor window and broke, Ryan would live on the first floor. However if it did not break, it would have to be dropped from all subsequent floors until it broke. So the *worst* case minimum number of drops for *one* vase (**V**=1) and an **N** story building is always **N** drops. It turns out that if Ryan's drop scheme is used, the *worst* case minimum number of drops falls off rapidly as the number of vases increases. For example, only 14 drops are required (worst case) for a 100 story building if two (**V** = 2) vases are used, and the worst case number of drops decreases to 9 if three vases are used.

Your task is to discover Ryan's (the optimum) drop algorithm, and then code it into a program to determine the minimum number of drops (worst case) given the number of vases Ryan purchased, **V**, and the number of floors in the building, **N**. Assume that if a vase is dropped and it does not break, it can be reused.

Inputs:
The first line of input contains the number of cases to consider. This will be followed by one line of input per case. Each line will contain two integers: the number of vases Ryan purchased, **V**, followed by the number of stories in the building, **N**.

Output:
There will be one line of output per case, which will contain the minimum number of drops, worst case.

Sample Input	**Sample Output**
5	(1 for each input)
2 10	4
2 100	14
3 100	9
25 900	10
5 100	7