Card Shuffling

Here is an algorithm for shuffling N cards:

1. The cards are divided into K equal piles, where K is a factor of N.
2. The bottom N / K cards belong to pile 1 in the same order (so the bottom card of the initial pile is the bottom card of pile 1).
3. The next N / K cards from the bottom belong to pile 2, and so on.
4. Now the top card of the shuffled pile is the top card of pile 1. The next card is the top card of pile 2,..., the K-th card of the shuffled pile is the top card of pile K. Then (K + 1)-th card is the card which is now at the top of pile 1, the (K + 2)-nd is the card which is now at the top of pile 2 and so on.

For example, if N = 6 and K = 3, the order of a deck of cards "ABCDEF" (top to bottom) when shuffled once would change to "ECAFDB".

Given N and K, what is the least number of shuffles needed after which the pile is restored to its original order?

Input

The first line contains the number of test cases T. The next T lines contain two integers each N and K.

Output

Output T lines, one for each test case containing the minimum number of shuffles needed. If the deck never comes back to its original order, output -1.

Constraints

T <= 10000
2 <= K <= N <= 10^9
K will be a factor of N.

Example

Sample Input:
3
6 3
4 2
5 5

Sample Output:
6
4
2