Optimal Swap Sort

One day, Budi is learning selection sort, He like this sort because the *swap* operation is always less than **n** where **n** is number of element to be sorted. Budi is smart person, so he modify the algorithm and make number of swap is as small as possible (Optimal). Now He want your help to analyze his algo, He wonder what is the worst number of swap operation needed when His algo sort **n** element(s) with **k** different element(s).

Input

The first line there is an integer **T** denoting number of test case.

Next **T** lines there are two integers **n** and **k** for each line, **n** and **k** has been described above.

Output

For each test case, output an integer that is the desired answer that has been described above.

Constraints

$1 \leq \mathbf{T} \leq 10^5$

$1 \leq \mathbf{k} \leq 10^9$

$\mathbf{k} \leq \mathbf{n} \leq 10^9$

Example

Input:

3

1 1

2 1

2 2

Output:

0

0

1

Explanation

For first and second test case it has one different element so it already sorted, no need to do any swap operation.

For last (third) test case if the element is decreasing (this is the worst case), only one swap is needed, then the sort is complete.

Other Info

Score is your source length.

Seems that it's impossible to solve this problem using some programming languages e.g(Brainf**k, Intercal, etc).

Click here to see my score for this problem (Link has been updated after rejudge)

See also: Another problem added by Tjandra Satria Gunawan