The first thing to do is to rewrite the original divisibility condition as $ab - n = p(a - n)(b - n)$, where $p > 1$ is a natural number. Suppose we fix a. After opening the brackets, we can get an expression for b: $b = \frac{n(p(a-n)-1)}{p(a-n)-a} = n + \frac{n(a-1)}{p(a-n)-a}$.

So, the first not-so-naive solution might be as follows. Check every possible value of a. Try every possible divisor of $n(a-1)$ (say d) to be the fraction's denominator (if the denominator is not a divisor of the numerator, b will not be an integer). As $p(a-n)-a = d$, we have $p = \frac{d+a}{a-n}$. So, in case $d+a$ is divisible by $a-n$ (remember that p must be an integer as well), a pair $(a, n + \frac{n(a-1)}{d})$ appears.

Of course, this approach is too slow, since there can be up to 10^{18} possible values of a. A good idea is to consider only pairs having $a \le b$ – we won't miss a pair, since if (a, b) is one of the sought pairs, then (b, a) is a sought pair too! Next, note that $d + a \ge 2(a - n)$ (since $p \ge 2$), which can be rewritten as $d \ge a - 2n$. It follows that if $a > 2n$, then $b \le n + \frac{n(a-1)}{a-2n}$, and if $a \le b$, then $a^2 - 4na + (n + 2n^2) \le 0$. Solving this, we get $a \le 2n + \sqrt{2n^2 - n}$. It's an important conclusion: instead of checking every possible value of a up to k, it's enough to check only values of a less than roughly $3.42n$. For the contestants, though, the exact bound was not needed, since it's possible to guess the limit or just code a bruteforce solution to see that.

So, now we can try every value of a. The last thing we need is to find all the divisors of $n(a-1)$ quickly. There are two main ways to do that:

1) Using the Sieve of Eratosthenes, precalculate the smallest prime divisor of every integer up to 342000 and then every integer's prime factorization. As both n and $a-1$ are smaller than 342000, it's now easy to find the prime factorization of $n(a-1)$ since we know the prime factorizations of n and $a-1$. Then it's easy to find all the divisors of $n(a-1)$ using a small recursive procedure.

2) Precalculate not just prime factorizations, but the list of divisors of every integer. Then every divisor of $n(a-1)$ is a product of a particular divisor of n and a particular divisor of $a-1$, so just two loops are enough. This approach is easier to code but slower, since we may find the same divisor of $n(a-1)$ more than once.

We came to an approach which works in a reasonable amount of time, but still several times too slow. Let's think: how many possible divisors d of $n(a-1)$ exist such that $d + a$ is divisible by $a - n$? In fact, it's not very hard to see that this number decreases a lot as a increases. This means that instead of checking every d, we might check every p which can be faster in some cases.

From $a \le b$, that is, $a \le n + \frac{n(a-1)}{p(a-n)-a}$ we can get $p \le \frac{a^2-n}{(a-n)^2}$. Note that for $a \ge n + 3000$ the upper bound on p is at most 1177 (for $n = 100000$). This number is not really big, and becomes even smaller as a increases. Actually, if for $a \ge n + 3000$ we loop over p and not over d, the solution becomes the needed number of times faster. This is enough to pass the time limit, though might need a bit of optimization.

And a few tips for the beginners:

1) Data type "long long" in C++, "long" in Java can hold numbers up to $2^{63} - 1 > 9 * 10^{18}$. If you want to read a variable of this type in C++ (at least at CodeChef), use cin (works everywhere, but works slower) or scanf with "%lld" specifier.

2) About $10^8 - 10^9$ simple operations can be done in a second on the modern computers. This problem thus can't be solved by two simple loops, since it would need about 10^{36} operations in the worst case, which is way too much.