This is a [reverse challenge](keyword://reverse-challenge), enjoy!

* **[time limit] 3000ms (java)**
* **[input] integer number**

*Guaranteed constraints:*  
0 ≤ number ≤ 109.

* **[output] string**

<https://codefights.com/challenge/xAfWeNKqLqeerhKXy?utm_source=emailNotification&utm_medium=email&utm_campaign=featuredChallenge>

static String iDontKnow(int number)

{

long prod = ((long)number \* (long)number);

prod -= (float)number;

return String.valueOf(prod);

}

-----------------------------------

[hydralisk](https://codefights.com/profile/hydralisk)

***Spoiler below***

The magic test case here is number = 147083927. Working with floating point gives a result that is 2 less than the mathematically correct result. If the intent of the challenge is for us to recognize that squaring large numbers results in floating point imprecision, then that's fine, but only if at least one of these two conditions is true:

1. We can determine the nature of the problem from the sample tests
2. The function name is a sufficient hint (e.g. it's named "float" or "imprecision" or something of that nature)

Since neither of these are true, this is not a well formulated reverse challenge. I'm inclined to agree with [@AWice](https://codefights.com/profile/AWice) - the quality of reverse challenges seems to vary wildly.

**2upvotes - 0 downvotes**