Time Complexity: Primality

A *prime* is a natural number *greater than* 1 that has no positive divisors other than 1 and itself. Given p integers, determine the primality of each integer and print whether it is `Prime` or `Not prime` on a new line.

Note: If possible, try to come up with an $\mathcal{O}(\sqrt{n})$ primality algorithm, or see what sort of optimizations you can come up with for an $\mathcal{O}(n)$ algorithm. Be sure to check out the *Editorial* after submitting your code!

Function Description

Complete the *primality* function in the editor below. It should return **Prime** if n is prime, or **Not prime**.

primality has the following parameter(s):

- *n*: an integer to test for primality

Input Format

The first line contains an integer, p, denoting the number of integers to check for primality.
Each of the p subsequent lines contains an integer, n, the number you must test for primality.

Constraints

- $1 \leq p \leq 30$
- $1 \leq n \leq 2 \times 10^9$

Output Format

For each integer, print whether n is `Prime` or `Not prime` on a new line.

Sample Input

```
3
12
5
7
```

Sample Output

```
Not prime
Prime
Prime
```

Explanation

We check the following $p = 3$ integers for primality:

1. $n = 12$ is divisible by numbers other than 1 and itself (i.e.: 2, 3, 6), so we print `Not prime` on a new line.
2. $n = 5$ is only divisible 1 and itself, so we print `Prime` on a new line.
3. $n = 7$ is only divisible 1 and itself, so we print `Prime` on a new line.