**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 O N  primality algorithm, or see what sort of optimizations you can come up with for an  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, , 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<=p<=30
* 1<=n<2X10^9

**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. 12 is divisible by numbers other than  and itself (i.e.: 2,3 ,4 ,6 ), so we print Not prime on a new line.
2. 5 is only divisible 1 and itself, so we print Prime on a new line.
3. 7 is only divisible 1 and itself, so we print Prime on a new line.
4. using System.CodeDom.Compiler;
5. using System.Collections.Generic;
6. using System.Collections;
7. using System.ComponentModel;
8. using System.Diagnostics.CodeAnalysis;
9. using System.Globalization;
10. using System.IO;
11. using System.Linq;
12. using System.Reflection;
13. using System.Runtime.Serialization;
14. using System.Text.RegularExpressions;
15. using System.Text;
16. using System;
17. class Solution {
18. // Complete the primality function below.
19. private static Boolean isPrime(int n)
20. {
21. if (n < 2) return false;
22. if(n==2||n==3||n==5||n==7) return true;
23. if(n%2==0||n%3==0||n%5==0 ||n%7==0) return false;
24. for (var i = n/7+1; i > 1; i--)
25. {
26. if (n % i == 0)
27. return false;
28. }
29. return true;
31. }
32. // Complete the primality function below.
33. static string primality(int n)
34. {
35. if (isPrime(n))
36. {
37. return "Prime";
38. }
39. else
40. {
41. return "Not prime";
42. }
44. }
45. static void Main(string[] args) {
46. TextWriter textWriter = new StreamWriter(@System.Environment.GetEnvironmentVariable("OUTPUT\_PATH"), true);
47. int p = Convert.ToInt32(Console.ReadLine());
48. for (int pItr = 0; pItr < p; pItr++) {
49. int n = Convert.ToInt32(Console.ReadLine());
50. string result = primality(n);
51. textWriter.WriteLine(result);
52. }
53. textWriter.Flush();
54. textWriter.Close();
55. }
56. }

**Congratulations**

You solved this challenge. Would you like to challenge your friends?

[Next Challenge](https://www.hackerrank.com/challenges/friend-circle-queries?h_l=interview&playlist_slugs%5B%5D=interview-preparation-kit&playlist_slugs%5B%5D=miscellaneous&h_r=next-challenge&h_v=zen)

* **Test case 0**
* **Test case 1**
* **Test case 2**
* **Test case 3**
* **Test case 4**
* **Test case 5**
* **Test case 6**
* **Test case 7**
* **Test case 8**
* **Test case 9**
* **Test case 10**

Compiler Message

**Success**

Input (stdin)

Download

* **3**
* **12**
* **5**
* **7**

Expected Output

Download

* **Not prime**
* **Prime**
* **Prime**