SWE2004: Principles in Programming(Spring 2014)

Programming Lab #7

Due-date: May 1, 14:45 PM

Q&A: jminlee at csl.skku.edu

Description

Certain cryptographic algorithms make use of big prime numbers. However, checking whether a big number is prime is not so easy.

Randomized primality tests exist that offer a high degree of confidence of accurate determination at low cost, such as the Fermat test. Let a be a random number between 2 and $n-1$, where n is the number whose primality we are testing. Then, n is probably prime if the following equation holds:

$$a^n \bmod n = a$$

If a number passes the Fermat test several times, then it is prime with a high probability.

Unfortunately, there is bad news. Certain composite numbers (non-primes) still pass the Fermat test with every number smaller than themselves. These numbers are called Carmichael numbers.

Write a program to test whether a given integer is a Carmichael number.

Input

The input will consist of a series of lines, each containing a small positive number n ($2 < n < 65000$). The first line of input containing a positive number T, which is the number of input.

Output

For each number in the input, print whether it is a Carmichael number or not as shown in the sample output.

Sample Iuput

```
5
1729
17
561
1109
431
```

Sample Output

```
The number 1729 is a Carmichael number.
17 is normal.
The number 561 is a Carmichael number.
1109 is normal.
431 is normal.
```

Reference

an efficient method to calculate $a^n \bmod n = a$

http://en.wikipedia.org/wiki/Modular_exponentiation