COT 3100 Program #3 Spring 2014
Assigned: 2/24/2014
Check WebCourses for due date

Note: This program (and all programs for the course) are ONLY for students who signed up for the programming option. If you signed up for this option, you MUST DO ALL FIVE programs assigned.

General Program Directions (to be followed for all five programs)

Turn in a single source file, either C or Java, with the name designated in the note at the bottom that solves the problem described below. Please read all of your input from standard in and output to standard out. To test your program with input files, please pipe the input file into your program and pipe the output to another file. If you don't know how to do this, please see a TA to describe this process to you.

The Problem: Calculating the Euler Phi Function

One of the key calculations in RSA encryption is the Euler Phi function, φ. In short, φ(n) is equal to the number of values from the set {1, 2, 3, ..., n} that are relatively prime with n. Two integers are relatively prime if their greatest common divisor is 1. Euler proved that for each positive integer a such that gcd(a, n) = 1, $a^{\phi(n)} \equiv 1 \ (mod\ n)$. It is this very fact that RSA encryption exploits. It turns out that if we have the prime factorization of an integer n, it's fairly easy to compute φ(n).

In particular, given that $n = \prod_{i=1}^{\infty} {p_i}^{a_i}$, we can calculate $\phi(n) = \prod_{i=1}^{\infty}({p_i}^{a_i} - {p_i}^{a_i-1})$.

For example, if n = $(2^4)(3)(17^3)$, then φ(n) = $(2^4 - 2^3)(3 - 1)(17^3 - 17^2)$.

For this problem, you will be asked to calculate φ(n) for values up to n = 10^{12}.

Problem Solving Outline

My aim here is for you to solve this problem in a specific way. The outline of the solution follows below. Please follow this outline. If you are curious as to why it's efficient, please come and ask me individually.

1. Precompute a list of all primes less than or equal to 10^6, using the Sieve or Eratosthenes. In particular, first use a boolean array and mark which items in it are prime. After finishing the sieve, go back through the array and count how many of the values in the range are prime. Then, create a new array of integers of exactly this size. Store each prime number in this new array. As an example, if we were only calculating primes up to 10, then the first part would use a boolean array of size 11, which would end up storing true in indexes 2, 3 5 and 7. The second part would copy these four values, 2, 3, 5 and 7 into a new array of size 4, so that primes[0] = 2, primes[1] = 3, primes[2] = 5 and primes[3] = 7.

2. For each case, do the following:

a. Read in the input.
b. Prime factorize the input value, n, as follows:
    i. Run through prime list, in order, trying to divide into n.
    ii. For each prime, p that divides evenly, reset n = n/p. Note this prime/power pair and continue.
    iii. Otherwise, go to the next prime.
c. If the leftover number is greater than 1, its prime, add it to the factorization.
d. Use the prime factorization of n and the formula for φ(n) to determine the answer.

Input Format

The first line of the input will contain a single positive integer, *c (c ≤ 10000)*, representing the number of input cases. The following *c* lines will each contain a single positive integer, *n (2 ≤ n ≤ 10^{12})* for which you are to calculate φ(n).

Output Format

For each input case, output φ(n) on a line by itself.

Sample Input

```
5
2
1000000000000
420107
2147483648
69984
```

Sample Output

```
1
400000000000
403200
1073741824
23328
```

Deliverables

A single source file, named either ***phi.c*** or ***phi.java*** that solves the program stated above, using the input and output formats stated above, using ***standard input and standard output.*** The file should be submitted via WebCourses by the due date and time stated in WebCourses.