KING SAUD UNIVERSITY
COLLEGE OF COMPUTER & INFORMATION SCIENCES
DEPT OF COMPUTER SCIENCE

CSC281 Discrete Mathematics for Computer Science Students
Second Semester 1441/1442 AH
Due: TBA
Instructor: Prof. Aqil Azmi

Group Term Project

In this project you will calculating the last few digits of a large integer, e.g. $2017^{2018^{2019}}$. You are all familiar with Fermat's Little theorem. For any prime p and an integer a such that $p \nmid a$, then $a^{p-1} \equiv 1 \bmod p$. There is a more general theorem known as Euler's theorem.

Euler's theorem. If a and n are relatively prime, then $a^{\phi(n)} \equiv 1 \bmod n$, where ϕ is Euler's totient function. It is defined as the number of integers between 1 and n that are relatively prime to n. In other words,

$$\phi(n) = \left|\{x \mid 1 \le x < n \;\wedge\; \gcd(n,x) = 1\}\right|.$$

For any prime p, we have $\phi(p) = p - 1$. We can see now why Fermat's Little theorem is a special case of Euler's theorem. We can easily compute the Euler's totient function as follows,

$$\phi(n) = n\prod_{p|n}(1 - 1/p) = n\prod_{p|n}\left(\frac{p-1}{p}\right),$$

where p is prime. For example, $\phi(12) = 12 \cdot (1 - 1/2) \cdot (1 - 1/3) = 4$, and $\phi(20) = 8$. For example, applying Euler's theorem we get $5^4 \equiv 1 \bmod 12$, and $5^{1000} \equiv 1 \bmod 12$. The latter is true since $5^{1000} = (5^4)^{250}$.

Back to our original problem. If we want the last two digits of $2017^{2018^{2019}}$ then we want to compute $2017^{2018^{2019}} \bmod 100$. Similarly, if we wanted the last three digit of the same, then it is a matter of computing $2017^{2018^{2019}} \bmod 1000$.

Let $x = 2018^{2019}$, so $2017^{2018^{2019}} \bmod 100 \equiv 2017^x \bmod 100 \equiv 17^x \bmod 100$. Why? Because,

$$2017^x \bmod 100 \equiv \underbrace{(2017 \bmod 100) \times \cdots \times (2017 \bmod 100)}_{x \text{ times}} \equiv (17 \bmod 100)^x \bmod 100 \equiv 17^x \bmod 100.$$

If we apply Euler's theorem we have, $17^{\phi(100)} = 17^{40} \equiv 1 \bmod 100$. Recall our $x = 2018^{2019}$, so we have $x \bmod \phi(100) \equiv x \bmod 40 \equiv 2018^{2019} \bmod 40 \equiv (2018 \bmod 40)^{2019} \equiv 18^{2019} \bmod 40$. Unfortunately we can't calculate $18^{2019} \bmod 40$ since gcd(18, 40) ≠ 1. Use the fast algorithm pm40 (see below) to calculate $d^e \bmod 40$ efficiently, where d and e are positive integers. In our case $d = 18$, and $e = 2019$.

```
procedure pm40(d,e) {
    if (e = 0) return 1
    x ← 1;  y ← d
    while (e > 0) do {
        if (e is odd) x ← x · y mod 40
        y ← y · y mod 40
        e ← ⌊e/2⌋
    }
    return x
}
```

Calculating pm40(18, 2019) we get 32. Thus, $2017^{2018^{2019}} \bmod 100 \equiv 17^{32} \bmod 100 \equiv 61$. Therefore the last two digits of $2017^{2018^{2019}}$ is 61. The whole number is ~ 26,700 digits long. One final note. To calculate $17^{32} \bmod 100$ you can use the algorithm above and replace all mod 40 by mod 100.

Project

Write a program that accepts four inputs: *a*, *b*, and *c*. Your program should return the *last two digits* of the number *a*^(*b*^*c*), that is a^{b^c}.

Test Examples

Test your program on the following data set. 2017^(2018^2019); 2018^(2019^2020); 2019^(2020^2021); 786^(786^786); and 1995^(1997^2000).

Instructions

This is a group project. Each 4 students will work as a team. You are free to use *any* convenient programming language. This project is worth 15 points.

What to submit

(a) Cover sheet with your names, class attendance number, and a signed pledge.
(b) Write-up of the project (brief description of your algorithm; the data structure(s) used; sample runs and the conclusion).
(c) Hardcopy of your source code + Flash memory/CD with source and executable.