NTUST OOP Midterm Problem Design

Subject: Square and Multiply

Author: 廖聖郝 (SHENG-HAO LIAO)

Main testing concept: 大數運算、二進制轉換、演算法

Basics

■ C++ BASICS
■ FLOW OF CONTROL
□ FUNCTION BASICS
□ PARAMETERS AND OVERLOADING
□ ARRAYS
□ STRUCTURES AND CLASSES
□ CONSTRUCTORS AND OTHER TOOLS
□ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES
□ STRINGS
□ POINTERS AND DYNAMIC ARRAYS

Functions

□ SEPARATE COMPILATION AND NAMESPACES
□ STREAMS AND FILE I/O
□ RECURSION
□ INHERITANCE
□ POLYMORPHISM AND VIRTUAL FUNCTIONS
□ TEMPLATES
□ LINKED DATA STRUCTURES
□ EXCEPTION HANDLING
□ STANDARD TEMPLATE LIBRARY
□ PATTERNS AND UML

Description:

$$47^{8943793798137911527249106497563} \mod 159 = ?$$

In computers, due to the issue of large numbers and performance, we cannot use regular calculation methods to handle the above equation. However, there is a common algorithm called "Square and Multiply" that can quickly calculate $a^b \mod p$ (b is large number).

$$a^b \mod p$$

Square and Multiply Algorithm:

1. Transfer large number b into binary format.
   e.g.: $20_{(10)}$ → $10100_{(2)}$,
   $8943793798137911527249106497563_{(10)}$ →
   $111000011100010111101001101010100011110001011001101110001110101111010001111$
   $1010010110001111000000011011_{(2)}$
2. You have a variable called *Result*, and initial with value 1.
3. Scan the binary from left to right, and do:
   (1) If digit is 1 execute <square>, then execute <multiply>
   (2) If digit is 0 execute <square>
   e.g.: $10100_{(2)}$ → $\underbrace{\text{<square><multiply>}}_{1}, \underbrace{\text{<square>}}_{0}, \underbrace{\text{<square><multiply>}}_{1}, \underbrace{\text{<square>}}_{0}, \underbrace{\text{<square>}}_{0}$
4. After all iteration *Result* variable is the value of $a^b \mod p$

Operations:
<square>: square *Result* and mod p, then store back into *Result*.
<multiply>: multiply *Result* by a and mod p, then store back into *Result*.

Input:

Each test cases may contain multiple inputs and outputs. Each line contains three positive integers a, b and p (a < 100000, b is a very large number that cannot be represented by a built-in data type, p < 10000).

a b p

Output:

Value of $a^b \ mod \ p$ for every inputs.

Sample Input / Output：

Sample Input	Sample Output

2 5 7
3 45 7
2447 5992 873
7414
898989898989898989898
9898989 9453
47
894379379813791152724
9106497563 159
	4
	6
	1
	4537
	119

■ **Eazy,Only basic programming syntax and structure are required.**

□ **Medium,Multiple programming grammars and structures are required.**

□ **Hard,Need to use multiple program structures or more complex data types.**

Expected solving time:

20 minutes

Other notes:

This is an algorithm commonly used in asymmetric encryption/decryption.