6 Homework Assignment

6.1 Integer Power

In cryptographic methods such as RSA, the integer power operation plays an important role. For positive integers a and m, the value $a^m \bmod N$ has to be calculated, where, $\bmod$ is the modulo operator that takes the remainder after division by N (`%` in Java).

1. Since m can be very large, simply repeatedly multiplying by a does not suffice. Therefore, use the property $a^{2m} = (a^2)^m$ to define a recurrence relation and base a recursive implementation of a function `powrec` upon this that uses fewer multiplications. This function calculates $a^m \bmod N$ for given $a \geq 1$, $m \geq 0$, and $N \geq 2$. N can be an instance variable or a parameter of the function (your choice).

   Your program should ask for a value of N and then repeatedly ask for values for a and m and print the value of $a^m \bmod N$. If N or a is too large or too small to use for your program, give a message and ask for a new value. Your program should work for $0 \leq m \leq$ *Integer.MAX_VALUE* and you may assume that m has a value in this range.

   You may assume that all input numbers are within the range of `long`.

2. Give a calculation for the number of multiplications that your program needs to calculate $a^m \bmod N$. Your answer should be exact, but not necessarily in the form of a formula. A description in Dutch or English will do as well. If you can't give an exact answer, give an approximation or an upper and a lower bound.

   Add this calculation to your submission as text or PDF in a separate file.

3. Write a function `powit` that implements the power operation in an iterative manner, i.e., without using recursion, with roughly the same efficiency.

4. Make sure that no overflow occurs during all calculations of your program. An overflow occurs when the correct result of an arithmetic operation exceeds the bounds of the numeric type. Note that the actual, but incorrect, result will be within the bounds of the type and that no warning or error will be issued. E.g., 2147483647+1 results without ado in -2147483648.

   Therefore, choose proper bounds for the values of a and N and proper types (int or long) for your variables.

   Note that arithmetic *modulo N* and ordinary arithmetic share most of the laws. In particular, $x \cdot y \bmod N = (x \bmod N) \cdot (y \bmod N) \bmod N$.

5. Provide pre- and postconditions with your methods (except `main`), as well as "modifies" and "uses" clauses where appropriate and provide invariants with your loops.

Notes about RSA

In the cryptographic method RSA, which is used in many secure applications, such as https, messages are encoded as large integers and then raised to some power e to encrypt them. This power e is called the *public key*. ("Public" because this number is not secret and freely distributed). Decrypting a message amounts to taking the root of the power e modulo N, but this is difficult to do efficiently. This is because the power operation behaves rather chaotic.

6. Illustrate this chaotic behaviour by drawing an ASCII-plot of 2^m. Take $N = 77$ for example and make the width of your console window at least this size. Let m run from 0 to N and draw on line number m the value of $2^m \bmod N$ as an asterisk shifted that many positions to the right. The first few values will be orderly increasing (1, 2, 4, 8, etc.), but after you pass N, chaos will set in.

   The root operation can be calculated efficiently, however, when we know more about N. If we know the divisors of N, we can find a number d with the property that $(a^e)^d \bmod N = a$ for any a. So raising to the power d effectively takes the root of power e and inverts the encryption. If one doesn't know the divisors of N, d can only be found by exhaustive search, which will take very long when N is large. Therefore, d is kept secret and is called the private key. Furthermore, the divisors of N have to be very large and kept secret. N is chosen as the product of two very big prime numbers, with some additional properties such as not being equal or close to each other. Cracking RSA is tantamount to factorising large numbers and for the latter problem no efficient algorithms are known.

Submission

Submit a working program that demoes the methods you have written, without using input. The program will be run by Peach so that the reviewers can inspect the output.