•1) Write an iterative definition of this algorithm (no recursion allowed).

```plaintext
Mul\_larusse(a,b)
sum:= 0
While b>0 do
    If (b is odd) Then sum:= sum+a; b:= b-1
    b:= b/2
    a:= a+a
Return sum
```

•2) Find the largest value of \( n \) of type long for which this method finds the correct answer for all \( a,b \) with \( 0 \leq a,b \leq n-1 \) and compare it with the direct Java expression \((a*b) \mod n\). Write a Java program to find the largest correct values for both methods.

The limit for \((a*b) \mod n\) is due to the calculation of \( a*b \) that will overflow as soon as
\[
(n-1)*(n-1) > Maxlong = 2^{63}-1
\]
or equivalently if
\[
 n > 1 + \sqrt{2^{63} - 1}.
\]
Finally \((a*b) \mod n\) will not overflow if \( n \leq 3037000500 \).

The limit for \texttt{MulMOD}(a,b,n) is due to the calculation of \( a+a \) that will overflow as soon as
\[
(n-1)+(n-1) > Maxlong = 2^{63}-1
\]
or equivalently if
\[
 n > Maxlong/2+1.
\]
This leads to
\[
 n > 2^{62}+1/2.
\]
Finally \texttt{MulMOD}(a,b,n) will not overflow if \( n \leq 4611686018427387904 \).

•3) Find a wise way to compute \((a+b) \mod n\) allowing you to go even further in part \( \textbullet \, 2 \).

Always compute \((a+b) \mod n\) as “If \((a-n+b) < 0 \) Then Return \( a+b \) Else Return \( a-n+b \)”. This way the sum is bounded by \(-n \leq (a-n+b) < n-1\) that never overflows unless \( n > \text{Maxlong} \).
4) Write a Java program to compute $a^b \mod n$ for $a, b, n$ of type `long` (with $0 \leq a, b \leq n-1$) and run it with the values

\[
\begin{align*}
  a & := 1274434408442 \\
  b & := 589394265617 \\
  n & := 1606818609763
\end{align*}
\]

The answer is 308250308250. To compute correctly, you needed to use `ExpMOD(a,b,n)` using `MulMOD(a,b,n)` to do the multiplications. Otherwise you were facing overflow problems using $a*b\%n$.

Let $d = 433371342353$. Run your program with the values $c, d, n$.

The answer is 1274434408442 ($= a$).