Finally, by the multiplicativity of φ, it is clear that $a^{\varphi(m)} \equiv 1 \ mod\ p^\alpha$ (simply raise both sides of $a^{\varphi(p^\alpha)} \equiv 1 \ mod\ p^\alpha$ to the appropriate power). Since this is true for each $p^\alpha || m$, and since the different prime powers have no common factors with one another, it follows by Property 5 of congruences that $a^{\varphi(m)} \equiv 1 \ mod\ m$.

Corollary. *If g.c.d.$(a, m) = 1$ and if n' is the least nonnegative residue of n modulo $\varphi(m)$, then $a^n \equiv a^{n'} \ mod\ m$.*

This corollary is proved in the same way as the corollary of Proposition I.3.2.

Remark. As the proof of Proposition I.3.5 makes clear, there's a smaller power of a which is guaranteed to give $1 \ mod\ m$: the least common multiple of the powers that give $1 \ mod\ p^\alpha$ for each $p^\alpha || m$. For example, $a^{12} \equiv 1 \ mod\ 105$ for a prime to 105, because 12 is a multiple of $3-1$, $5-1$ and $7-1$. Note that $\varphi(105) = 48$. Here is another example:

Example 3. Compute $2^{1000000} \ mod\ 77$.

Solution. Because 30 is the least common multiple of $\varphi(7) = 6$ and $\varphi(11) = 10$, by the above remark we have $2^{30} \equiv 1 \ mod\ 77$. Since $1000000 = 30 \cdot 33333 + 10$, it follows that $2^{1000000} \equiv 2^{10} \equiv 23 \ mod\ 77$. A second method of solution would be first to compute $2^{1000000} \ mod\ 7$ (since $1000000 = 6 \cdot 166666 + 4$, this is $2^4 \equiv 2$) and also $2^{1000000} \ mod\ 11$ (since 1000000 is divisible by $11-1$, this is 1), and then use the Chinese Remainder Theorem to find an x between 0 and 76 which is $\equiv 2 \ mod\ 7$ and $\equiv 1 \ mod\ 11$.

Modular exponentiation by the repeated squaring method. A basic computation one often encounters in modular arithmetic is finding $b^n \ mod\ m$ (i.e., finding the least nonnegative residue) when both m and n are very large. There is a clever way of doing this that is much quicker than repeated multiplication of b by itself. In what follows we shall assume that $b < m$, and that whenever we perform a multiplication we then immediately reduce $mod\ m$ (i.e., replace the product by its least nonnegative residue). In that way we never encounter any integers greater than m^2. We now describe the algorithm.

Use a to denote the partial product. When we're done, we'll have a equal to the least nonnegative residue of $b^n \ mod\ m$. We start out with $a = 1$. Let $n_0, n_1, \ldots, n_{k-1}$ denote the binary digits of n, i.e., $n = n_0 + 2n_1 + 4n_2 + \cdots + 2^{k-1}n_{k-1}$. Each n_j is 0 or 1. If $n_0 = 1$, change a to b (otherwise keep $a = 1$). Then square b, and set $b_1 = b^2 \ mod\ m$ (i.e., b_1 is the least nonnegative residue of $b^2 \ mod\ m$). If $n_1 = 1$, multiply a by b_1 (and reduce $mod\ m$); otherwise keep a unchanged. Next square b_1, and set $b_2 = b_1^2 \ mod\ m$. If $n_2 = 1$, multiply a by b_2; otherwise keep a unchanged. Continue in this way. You see that in the j-th step you have computed $b_j \equiv b^{2^j} \ mod\ m$. If $n_j = 1$, i.e., if 2^j occurs in the binary expansion of n, then you include b_j in the product for a (if 2^j is absent from n, then you do not). It is easy to see that after the $(k-1)$–st step you'll have the desired $a \equiv b^n \ mod\ m$.