# The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

*Solution Manual*
*Chapter 5, Problem 5.11*

1. Note that this function is only defined for $n > 0$: 0 itself does not have a prime power decomposition. We know from Exercise 5.5 that the function prime($i$), which returns the $i$th prime, is primitive recursive, as is the bounded maximization operator (see below), as is the predicate $x|y$. We can define the primitive recursive function power($i, n$), which raises $n$ to the $i$th power, as follows:

$$\begin{cases} \text{power}(0, n) = 1 \\ \text{power}(i + 1, n) = \text{mult}(n, \text{power}(i, n)) \end{cases}$$

(where we omitted the projection functions). An easy definition of exp($n, m$) can then be had by substitution:

$$\exp(n, m) = \max y \leq n[\text{power}(y, \text{prime}(m))|n]$$

2. We can define this function by primitive recursion; call the function $f(x, z_1, \ldots, z_n)$ and define:

$$\begin{cases} f(0, z_1, \ldots, z_n) = g(0, z_1, \ldots, z_n) \\ f(n + 1, z_1, \ldots, z_n) = \text{larger}(f(n, z_1, \ldots, z_n), g(n + 1, z_1, \ldots, z_n)) \end{cases}$$

(omitting the projections needed to make this perfectly legal), where larger is defined by substitution as

$$\text{larger}(x, y) = \text{con}_2(\text{lev}(\text{minus}(x, y))\#x, \text{is\_zero}(\text{minus}(x, y))\#y)$$

using much the same idea (concatenation of two arguments, one of which is always 0) as in the definition by cases of Exercise 5.3.

3. With the course-of-values recursion of Equation 5.1 (and Exercise 5.2), this problem is a breeze! The general definition just uses Equation 5.1 to write:

$$f(n+1) = \mathrm{add}\,(\Pi(2, n+1, \langle n+1, f(n), f(n-1), \ldots, f(0)\rangle), \Pi(3, n+1, \langle n+1, f(n), f(n-1), \ldots, f(0)\rangle))$$

   That is, the function $h$ of the definition is a function that adds the second and the third terms in the pairing and returns that sum. We do need to specialize the definition of Equation 5.1 to a function $f$ of just one variable, so that the base case invokes a function $g$ of zero variables—a constant. However, it is not quite enough to write

$$f(0) = 1$$

   since we need two base cases, not just one. But we can handle this problem rather easily through a definition by cases and an artificial shift of arguments.
   We can now give the correct definition. We define $f$ by cases:

$$f(n) = \begin{cases} 1 & n = 0 \\ f'(n-1) & \text{otherwise} \end{cases}$$

   and define $f'$ by course-of-values recursion, where the key function $p$ is slightly modified to $p'$, yielding

$$\begin{cases} f'(0) = 1 \\ f'(n+1) = h(n, p'(n)) \end{cases}$$

   The modified $p$ is defined by

$$\begin{cases} p'(0) = \langle 1, 1\rangle \\ p'(n+1) = \langle h(n, p'(n)), p'(n)\rangle \end{cases}$$

   so as to include the two base cases at once—and also simplified by omitting the index in the pairing (since we always know that value of that index). Finally, as we mentioned above, the $h$ adds the now first and second (because we dropped the index) terms of its pairing, so we define it as

$$h(i, j) = \mathrm{add}(\Pi(1, i+2, j), \Pi(2, i+2, j))$$

   (where the $i+2$ is due to the initial pairing of two base values).

This technique shows us how to handle the definition of recurrences of any constant order: just pack all the base cases into one pairing as the base case of a modified $p$, then define the main function by cases and by shifting the argument.