Question 1:

For each of the following functions, show that it is primitive recursive. Use a “relaxed” mathematical argument such as we did on slides 9 to 15 on September 28. You do not have to provide a precise proof as we did on slides 7 and 8, but you can if you like. In your arguments, you can use any functions that we have already shown in class to be primitive recursive without having to prove it again.

(a) \( f(x) = \lfloor \sqrt{x} \rfloor \)

(b) \( g(x_1, x_2) = \max(x_1 x_2, x_1 + x_2) \) (Here, max returns the larger of its two inputs).

(c) \( h(x) = \begin{cases} 0, & \text{if } \text{Prime}(x) \\ 1, & \text{if } \sim \text{Prime}(x) \& \text{Prime}(x + 1) \\ x + 1, & \text{otherwise} \end{cases} \)

(d) \( k(x) \) is the number of twin primes whose smaller twin is less than \( x \). Twin primes are pairs of primes whose difference is two. For example, \((3, 5)\) and \((11, 13)\) are twin primes.

(e) Predicate \( M(x) \) indicating whether \( x \) is a perfect number. Perfect numbers are those which are equal to the sum of all of their divisors except themselves. For example, 6 is a perfect number, because its divisors are 1, 2, and 3, and \( 1 + 2 + 3 = 6 \). Also, 28 is a perfect number, because \( 1 + 2 + 4 + 7 + 14 = 28 \).
Question 2:

Write an $\mathcal{L}$ program to compute each of the following functions. You can use macros but have to provide code for any functions or predicates used as macros. You can use the Haskell tools to test your code. You can write down the code in standard $\mathcal{L}$ notation or in the Haskell (.l) format.

(a) $Q(x, y) = x | y$ (as defined on slide 12 on October 3)

(b) Prime($x$) (as defined on slide 13 on October 3)

(c) $p_n$ (as defined on slide 19 on October 3)