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

This can be written as:

\[ f(x) = \min_{t \leq x} [(t + 1) \cdot (t + 1) > x] \]

We have already shown bounded minimalization, multiplication, and the “greater than“ function to be primitive recursive, so \( f(x) \) is also primitive recursive.

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

This can be written as:

\[ g(x_1, x_2) = \begin{cases} 
  x_1 x_2, & \text{if } x_1 x_2 > x_1 + x_2 \\
  x_1 + x_2, & \text{otherwise}
\end{cases} \]

We have already shown that addition, multiplication, “greater than” and case distinctions by predicates (“switch statements”) are primitive recursive, so this function must also be primitive recursive.

(c) \( h(x) = \begin{cases} 
  0, & \text{if Prime}(x) \\
  1, & \text{if } \sim \text{Prime}(x) \land \text{Prime}(x+1) \\
  x + 1, & \text{otherwise}
\end{cases} \)
Sorry, there is really nothing to do here. Since we already showed that Prime(x), the “and” operator, addition, and case distinctions are primitive recursive, \( h(x) \) is primitive recursive as well.

(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.

Let us first define a predicate IsSmallTwin:

\[
\text{IsSmallTwin}(x) = \text{Prime}(x) \land \text{Prime}(x + 2)
\]

Then we can compute \( k(x) \) recursively as follows:

\[
k(0) = 0 \\
k(t + 1) = k(t) + \text{IsSmallTwin}(t)
\]

Clearly, the functions we use are all primitive recursive, and then so is \( k(t) \).

(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 \( 1 + 2 + 4 + 7 + 14 = 28 \).

We can compute \( M(x) \) as follows:

\[
M(x) = \left( \sum_{i=1}^{x-1} i \cdot (i \mid x) \right) = x
\]

We already know summation, the “divides” predicate, multiplication and equality are primitive recursive, and therefore, \( M(x) \) is as well.

**Question 2:**

Write an \( L \) 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 \( L \) notation or in the Haskell (.l) format.

(a) \( Q(x, y) = x \mid y \) (as defined on slide 12 on October 3)

\[
\text{divides.l:}
\]

```plaintext
SET Z1 FUNC MODULO X2 X1
IF Z1 GOTO E
INC Y
```
modulo.l:

IF X2 GOTO A2
[A1]  GOTO A1
[A2]  IF X1 GOTO A3
      GOTO E
[A3]  INC Y
      SET Z1 FUNC ISEQUAL Y X2
      IF Z1 GOTO A5
[A4]  DBC X1
      GOTO A2
[A5]  SET Y VAL 0
      GOTO A4

isequal.l:

[A1]  IF X1 GOTO A2
      IF X2 GOTO E
      INC Y
      IF Y GOTO E
[A2]  IF X2 GOTO A3
      INC Z1
      IF Z1 GOTO E
[A3]  DEC X1
      DBC X2
      INC Z1
      IF Z1 GOTO A1

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

prime.l:

IF X1 GOTO A1
[A1]  DBC X1
      IF X1 GOTO A2
      GOTO E
[A2]  INC X1
      INC Z5
      INC Z5
      SET Z1 FUNC DIV X1 Z5
      INC Z2
[A3]  INC Z2
      SET Z3 FUNC ISGREATER Z2 Z1
      IF Z3 GOTO A4
      SET Z4 FUNC DIVIDES Z2 X1
IF Z4 GOTO E
GOTO A3
[A4] INC Y

div.l:

IF X2 GOTO A2
[A1] GOTO A1
[A2] IF X1 GOTO A3
 GOTO E
[A3] SET Z1 VAR X2
[A4] IF X1 GOTO A5
 GOTO E
[A5] DEC X1
 DEC Z1
 IF Z1 GOTO A4
 INC Y
 GOTO A3

isgreater.l:

[A1] IF X1 GOTO A2
 GOTO E
[A2] IF X2 GOTO A3
 INC Y
 GOTO E
[A3] DEC X1
 DEC X2
 GOTO A1

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

pn.l:

IF X1 GOTO A1
GOTO E
[A1] INC Y
 SET Z1 FUNC PRIME Y
 IF Z1 GOTO A2
 GOTO A1
[A2] DEC X1
 IF X1 GOTO A1