Question 1:

For each of the following functions, show that it is primitive recursive. Use a “relaxed” mathematical argument such as we did starting on slide 7 on February 26. You do not have to provide a precise proof as we did on the earlier slides, 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) \( \text{gcd}(0, 0) = 0 \)
\( \text{gcd}(x_1, 0) = x_1 \)
\( \text{gcd}(0, x_2) = x_2 \)
\( \text{gcd}(x_1, x_2) = \text{gcd}(R(x_2, x_1), x_1) \)

We know that the remainder function \( R \) is primitive recursive, and even though the form of recursion on it used here is different from the one discussed in class, it is sufficient to show that \( \text{gcd} \) is a primitive recursive function.

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

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.

(c) \( g(x_1, x_2) = \min(x_1^2, x_2, x_1x_2^2) \) (Here, \( \min \) returns the smaller of its two inputs).

This can be written as:

\[
 g(x_1, x_2) = \begin{cases} 
 x_1^2, & \text{if } x_1^2 < x_1x_2^2 \\
 x_1x_2^2, & \text{otherwise}
\end{cases}
\]
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.

(d) $k(x_1, x_2)$ is the number of twin primes whose smaller twin is greater than $x_1$ and less than $x_2$. 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:

$$\text{IsSmallTwin}(x) = \text{Prime}(x) \& \text{Prime}(x + 2)$$

Then we can compute $k(x_1, x_2)$ as follows:

$$k(x_1, x_2) = \sum_{x=0}^{x_1} \text{IsSmallTwin}(x) - \sum_{x=0}^{x_2} \text{IsSmallTwin}(x)$$

Clearly, the functions we use are all primitive recursive, and then so is $k(x_1, x_2)$.

**Question 2:**

Write an \textit{L} program to compute each of the following functions as they are defined on the slides. 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 \textit{L} notation or in the Haskell (.l) format.

(a) $Q(x, y) = x \mid y$

\texttt{divides.1}:

```l
SET Z1 FUNC MODULO X2 X1
IF Z1 GOTO E
INC Y
```

\texttt{modulo.1}:

```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] DEC X1
```
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
    DEC X2
    INC Z1
    IF Z1 GOTO A1

(b) Prime(x)

prime.l:

    IF X1 GOTO A1
    GOTO E
[A1] DEC 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) $p_n$

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