Question 1:

(a) Let $f(x) = 1$ if $x$ is even, $f(x) = 2$ if $x$ is odd. Show that $f(x)$ is primitive recursive.

We can compute $f(x)$ as follows:

$$f(x) = 2 \text{ monus } (2 \mid x)$$

We have already shown that “monus” and “$|$” are primitive recursive, and therefore, $f(x)$ must also be primitive recursive.

(b) Show that $f(x)$ is partially computable by writing a program in language $L$ that computes $f(x)$. Do not use any macros but only the proper $L$ commands. By the way, this will allow you to test your program using the Haskell code.

```
Y ← Y + 1
IF X ≠ 0 GOTO A
IF Y ≠ 0 GOTO E
[A] X ← X - 1
IF X ≠ 0 GOTO B
Y ← Y + 1
IF Y ≠ 0 GOTO E
[B] X ← X - 1
IF X ≠ 0 GOTO A
```

Question 2:

Let $g(x) = 2x$ if $x$ is a perfect square, $g(x) = 2x + 1$ otherwise. Show that $g(x)$ is primitive recursive.

We can compute $g(x)$ as follows:

$$g(x) = 2x \cdot (\exists t)x.t \cdot t = x + (2x + 1) \cdot \alpha[(\exists t)x.t \cdot t = x]$$
Since we have already shown the “exist” predicate, multiplication, addition, and $\alpha$ to be primitive recursive, $g(x)$ must also be primitive recursive.

**Question 3:**

Let $h(x)$ be the number of primes that are less than $x$. Show that $h(x)$ is primitive recursive.

We can compute $h(x)$ as follows:

$$h(x) = \min_{t \leq x} (p_{t+1} \geq x)$$

We have already shown bounded minimalization, “greater than or equal,” and addition to be primitive recursive, and therefore, $h(x)$ is also primitive recursive.

**Question 4:**

Let $k(x)$ be the integer $n$ such that $n \leq \sqrt{2x} \leq n + 1$. Show that $k(x)$ is primitive recursive.

In other words, we are looking for the largest integer $n$ so that $n \leq \sqrt{2x}$. If we compute the square on both sides, we get:

$$n^2 \leq 2x^2$$

Therefore, $k(x)$ can be computed as follows:

$$k(x) = \min_{t \leq x} [(t + 1) \cdot (t + 1) > 2x \cdot x]$$

We have already shown bounded minimalization, “greater than,” multiplication, and addition to be primitive recursive, and therefore, $k(x)$ is also primitive recursive.