We already have some examples of primitive recursive functions: the addition and multiplication functions \text{add} and \text{mult}. The identity function \text{id}(x) = x is primitive recursive, since it is just $P_1^1$. The constant functions $\text{const}_n(x) = n$ are primitive recursive since they can be defined from zero and \text{succ} by successive composition. This is useful when we want to use constants in primitive recursive definitions, e.g., if we want to define the function $f(x) = 2 \cdot x$ can obtain it by composition from $\text{const}_n(x)$ and multiplication as $f(x) = \text{mult} (\text{const}_2(x), P_1^1(x))$. We’ll make use of this trick from now on.

**Proposition rec.1.** The exponentiation function $\exp(x, y) = x^y$ is primitive recursive.

*Proof.* We can define $\exp$ primitive recursively as

\[
\begin{align*}
\exp(x, 0) &= 1 \\
\exp(x, y + 1) &= \text{mult}(x, \exp(x, y)).
\end{align*}
\]

Strictly speaking, this is not a recursive definition from primitive recursive functions. Officially, though, we have:

\[
\begin{align*}
\exp(x, 0) &= f(x) \\
\exp(x, y + 1) &= g(x, y, \exp(x, y)).
\end{align*}
\]

where

\[
\begin{align*}
f(x) &= \text{succ} (\text{zero}(x)) = 1 \\
g(x, y, z) &= \text{mult} (P_0^1(x, y, z), P_2^2(x, y, z)) = x \cdot z
\end{align*}
\]

and so $f$ and $g$ are defined from primitive recursive functions by composition.\qed

**Proposition rec.2.** The predecessor function $\text{pred}(y)$ defined by

\[
\text{pred}(y) = \begin{cases} 
0 & \text{if } y = 0 \\
y - 1 & \text{otherwise}
\end{cases}
\]

is primitive recursive.

*Proof.* Note that

\[
\begin{align*}
\text{pred}(0) &= 0 \\
\text{pred}(y + 1) &= y.
\end{align*}
\]

This is almost a primitive recursive definition. It does not, strictly speaking, fit into the pattern of definition by primitive recursion, since that pattern requires
at least one extra argument $x$. It is also odd in that it does not actually use \( \text{pred}(y) \) in the definition of \( \text{pred}(y+1) \). But we can first define \( \text{pred}'(x, y) \) by

\[
\begin{align*}
\text{pred}'(x, 0) &= \text{zero}(x) = 0, \\
\text{pred}'(x, y + 1) &= P_1^3(x, y, \text{pred}'(x, y)) = y.
\end{align*}
\]

and then define \( \text{pred} \) from it by composition, e.g., as \( \text{pred}(x) = \text{pred}'(\text{zero}(x), P_0^1(x)) \).

**Proposition rec.3.** The factorial function \( \text{fac}(x) = x! = 1 \cdot 2 \cdot 3 \cdots x \) is primitive recursive.

**Proof.** The obvious primitive recursive definition is

\[
\begin{align*}
\text{fac}(0) &= 1 \\
\text{fac}(y + 1) &= \text{fac}(y) \cdot (y + 1).
\end{align*}
\]

Officially, we have to first define a two-place function \( h \)

\[
\begin{align*}
h(x, 0) &= \text{const}_1(x) \\
h(x, y) &= g(x, y, h(x, y))
\end{align*}
\]

where \( g(x, y, z) = \text{mult}(P_2^3(x, y, z), \text{succ}(P_1^3(x, y, z))) \) and then let

\[
\text{fac}(y) = h(P_0^1(y), P_0^1(y))
\]

From now on we’ll be a bit more laissez-faire and not give the official definitions by composition and primitive recursion.

**Proposition rec.4.** Truncated subtraction, \( x \dot{-} y \), defined by

\[
x \dot{-} y = \begin{cases} 
0 & \text{if } x > y \\
 x - y & \text{otherwise}
\end{cases}
\]

is primitive recursive.

**Proof.** We have:

\[
\begin{align*}
x \dot{-} 0 &= x \\
x \dot{-} (y + 1) &= \text{pred}(x \dot{-} y)
\end{align*}
\]

**Proposition rec.5.** The distance between \( x \) and \( y \), \( |x - y| \), is primitive recursive.

**Proof.** We have \( |x - y| = (x \dot{-} y) + (y \dot{-} x) \), so the distance can be defined by composition from \( + \) and \( \dot{-} \), which are primitive recursive.
**Proposition rec.6.** The maximum of \( x \) and \( y \), \( \max(x, y) \), is primitive recursive.

**Proof.** We can define \( \max(x, y) \) by composition from \( + \) and \( \dot{-} \) by

\[
\max(x, y) = x + (y \dot{-} x).
\]

If \( x \) is the maximum, i.e., \( x \geq y \), then \( y \dot{-} x = 0 \), so \( x + (y \dot{-} x) = x + 0 = x \). If \( y \) is the maximum, then \( y \dot{-} x = y - x \), and so \( x + (y \dot{-} x) = x + (y - x) = y \). \( \square \)

**Proposition rec.7.** The minimum of \( x \) and \( y \), \( \min(x, y) \), is primitive recursive.

**Proof.** Exercise. \( \square \)

**Problem rec.1.** Prove Proposition rec.7.

**Problem rec.2.** Show that

\[
f(x, y) = 2^{(2^{x})^y 2^{x}}
\]

is primitive recursive.

**Problem rec.3.** Show that integer division \( d(x, y) = \lfloor x/y \rfloor \) (i.e., division, where you disregard everything after the decimal point) is primitive recursive. When \( y = 0 \), we stipulate \( d(x, y) = 0 \). Give an explicit definition of \( d \) using primitive recursion and composition.

**Proposition rec.8.** The set of primitive recursive functions is closed under the following two operations:

1. **Finite sums:** if \( f(\overline{x}, z) \) is primitive recursive, then so is the function

\[
g(\overline{x}, y) = \sum_{z=0}^{y} f(\overline{x}, z).
\]

2. **Finite products:** if \( f(\overline{x}, z) \) is primitive recursive, then so is the function

\[
h(\overline{x}, y) = \prod_{z=0}^{y} f(\overline{x}, z).
\]

**Proof.** For example, finite sums are defined recursively by the equations

\[
g(\overline{x}, 0) = f(\overline{x}, 0) \\
g(\overline{x}, y + 1) = g(\overline{x}, y) + f(\overline{x}, y + 1).
\]

\( \square \)