Generally useful information.

- The notation \( z = \langle x, y \rangle \) denotes the pairing function with inverses \( x = \langle z \rangle_1 \) and \( y = \langle z \rangle_2 \).

- The minimization notation \( \mu y \left[ P(\ldots, y) \right] \) means the least \( y \) (starting at 0) such that \( P(\ldots, y) \) is true. The bounded minimization (acceptable in primitive recursive functions) notation \( \mu y \left( u \leq y \leq v \right) \left[ P(\ldots, y) \right] \) means the least \( y \) (starting at \( u \) and ending at \( v \)) such that \( P(\ldots, y) \) is true. Unlike the text, I find it convenient to define \( \mu y \left( u \leq y \leq v \right) \left[ P(\ldots, y) \right] \) to be \( v + 1 \), when no \( y \) satisfies this bounded minimization.

- The tilde symbol, \(~\), means the complement. Thus, set \( \sim S \) is the set complement of set \( S \), and predicate \( \sim P(x) \) is the logical complement of predicate \( P(x) \).

- A function \( P \) is a predicate if it is a logical function that returns either 1 (true) or 0 (false). Thus, \( P(x) \) means \( P \) evaluates to true on \( x \), but we can also take advantage of the fact that true is 1 and false is 0 in formulas like \( y \times P(x) \), which would evaluate to either \( y \) (if \( P(x) \)) or 0 (if \( \sim P(x) \)).

- A set \( S \) is recursive if \( S \) has a total recursive characteristic function \( \chi_S \), such that \( x \in S \iff \chi_S(x) \). Note \( \chi_S \) is a predicate. Thus, it evaluates to 0 (false), if \( x \not\in S \).

- When I say a set \( S \) is re, unless I explicitly say otherwise, you may assume any of the following equivalent characterizations:
  1. \( S \) is either empty or the range of a total recursive function \( f_S \).
  2. \( S \) is the domain of a partial recursive function \( g_S \).

- If I say a function \( g \) is partially computable, then there is an index \( g \) (I know that’s overloading, but that’s okay as long as we understand each other), such that \( \Phi(g)(x) = \Phi(x, g) = g(x) \). Here \( \Phi \) is a universal partially recursive function.

  Moreover, there is a primitive recursive function \( STP \), such that \( STP(x, g, t) = 1 \) (true), just in case \( g \), started on \( x \), halts in \( t \) or fewer steps.

  Finally, there is another primitive recursive function \( VALUE \), such that \( VALUE(x, g, t) = g(x) \), whenever \( STP(x, g, t) \).

- The notation \( f(x)\downarrow \) means that \( f \) converges when computing with input \( x \), but we don’t care about the value produced. In effect, this just means that \( x \) is in the domain of \( f \).

- The notation \( f(x)\uparrow \) means \( f \) diverges when computing with input \( x \). In effect, this just means that \( x \) is not in the domain of \( f \).

- The **Halting Problem** for any effective computational system is the problem to determine of an arbitrary effective procedure \( f \) and input \( x \), whether or not \( f(x)\downarrow \). The set of all such pairs is a classic re non-recursive one.

- The **Uniform Halting Problem** is the problem to determine of an arbitrary effective procedure \( f \), whether or not \( f \) is an algorithm (halts on all input). The set of all such function indices is a classic non re one.