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

The tilde symbol, $\sim$, means the complement. Thus, set $\sim S$ is the set complement of set $S$, and the 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 total predicate. Thus, it evaluates to 0 (false), if $x \notin S$.

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 $g$ (we tend to overload the index as the function name), 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 $\text{STP}$, such that $\text{STP}(g, x, t)$ is 1 (true), just in case $g$, started on $x$, halts in $t$ or fewer steps.

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

The notation $f(x) \downarrow$ means that $f$ converges when computing with input $x$ ($x \in \text{Dom}(f)$). The notation $f(x) \uparrow$ means $f$ diverges when computing with input $x$ ($x \notin \text{Dom}(f)$).

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, $K_0$, is a classic re non-recursive set. $K_0$ is also known as $L_0$, the universal language. The related set, $K$, is the set of all effective procedures $f$ such that $f(f) \downarrow$ or more precisely $\Phi_{f(f)}(f)$.

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). This set, $\text{TOTAL}$, is a classic non re set.

When I ask for a reduction of one set of indices to another, the formal rule is that you must produce a function that takes an index of one function and produces the index of another having whatever property you require. However, I allow some laxness here. You can start with a function, given its index, and produce another function, knowing it will have a computable index. For example, given $f$, a unary function, I might define $G_f$, another unary function, by $G_f(0) = f(0)$; $G_f(y+1) = G_f(y) + f(y+1)$

This would get $G_f(x)$ as the sum of the values of $f(0)+f(1)+\ldots+f(x)$.

The Post Correspondence Problem (PCP) is known to be undecidable. This problem is characterized by instances that are described by a number $n > 0$ and two $n$-ary sequences of non-empty words $x_1, x_2, \ldots, x_n$, $y_1, y_2, \ldots, y_m$. The question is whether or not there exists a sequence, $i_1, i_2, \ldots, i_k$, such that $1 \leq i \leq n$, $1 \leq j \leq k$, and $x_1 x_2 \ldots x_k = y_1 y_2 \ldots y_k$.
• When I ask you to show one set of indices, $A$, is many-one reducible to another, $B$, denoted $A \leq_m B$, you must demonstrate a total computable function $f$, such that $x \in A \iff f(x) \in B$. The stronger relationship is that $A$ and $B$ are many-one equivalent, $A \equiv_m B$, requires that you show $A \leq_m B$ and $B \leq_m A$. The related notion of one-one reducibility and equivalence require that the reducing function, $f$ above, be 1-1. The notation just replaces the $m$ with a 1, as in $A \leq_1 B$. 
1. Let set $A$ be recursive, $B$ be re non-recursive and $C$ be non-re. Choosing from among (REC) recursive, (RE) re non-recursive, (NR) non-re, categorize the set $D$ in each of a) through d) by listing all possible categories. No justification is required.

   a.) $D = \sim C$
   b.) $D \subseteq (A \cup C)$
   c.) $D = \sim B$
   d.) $D = B - A$

2. Choosing from among (D) decidable, (U) undecidable, (?) unknown, categorize each of the following decision problems. No proofs are required.

<table>
<thead>
<tr>
<th>Problem / Language Class</th>
<th>Regular</th>
<th>Context Free</th>
<th>Context Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L = \Sigma^*$ ?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L = \phi$ ?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L = L^2$ ?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \in L^2$, for arbitrary $x$ ?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Use PCP to show the undecidability of the problem to determine if the intersection of two context free languages is non-empty. That is, show how to create two grammars $G_A$ and $G_B$ based on some instance $P = \langle \langle x_1, x_2, \ldots, x_n \rangle, \langle y_1, y_2, \ldots, y_n \rangle \rangle$ of PCP, such that $L(G_A) \cap L(G_B) \neq \phi$ iff $P$ has a solution. Assume that $P$ is over the alphabet $\Sigma$. You should discuss what languages your grammars produce and why this is relevant, but no formal proof is required.
4. Consider the set of indices \( \text{CONSTANT} = \{ f | \exists K \forall y \ [ \varphi_f(y) = K ] \} \). Use Rice’s Theorem to show that \( \text{CONSTANT} \) is not recursive. Hint: There are two properties that must be demonstrated.

5. Show that \( \text{CONSTANT} \equiv_m \text{TOT} \), where \( \text{TOT} = \{ f | \forall y \varphi_f(y) \downarrow \} \).
6. Why does Rice’s Theorem have nothing to say about the following? Explain by showing some condition of Rice’s Theorem that is not met by the stated property.

\[ \text{AT-LEAST-LINEAR} = \{ f | \forall y \varphi_y(f(y)) \text{ converges in no fewer than } y \text{ steps} \} \]

7. The trace language of a computational device like a Turing Machine is a language of the form

\[ \text{Trace} = \{ C_1 \# C_2 \# \ldots \# C_n \# | C_i \Rightarrow C_{i+1}, 1 \leq i < n \} \]

\text{Trace} is Context Sensitive, non-Context Free. Actually, a trace language typically has every other configuration word reversed, but the concept is the same. Oddly, the complement of such a trace is Context Free. Explain what makes its complement a CFL. In other words, describe the characteristics of this complement and why these characteristics are amenable to a CFG description.

8. We demonstrated a proof that the context sensitive languages are not closed under homomorphism, To start, we assumed \( G = (N, \Sigma, S, P) \) is an arbitrary Phrase Structured Grammar, with \( N \) its set of non-terminals, \( \Sigma \) its terminal alphabet, \( S \) its starting non-terminal and \( P \) its productions (rules). Since \( G \) is a PSG, it can have length increasing, length preserving and length decreasing rules. We wished to convert \( G \) to a CSG, \( G' = (N', \Sigma', S', P') \) where there are no rules that are length decreasing (since a CSG cannot have these). We developed a way to pad the length decreasing rules from \( G \) and then a homomorphism that gets rid of these padding characters. Define \( G' \) and the homomorphism \( h \) that we discussed in class and then briefly discuss why this new grammar and homomorphism combine so \( h(L(G')) = L(G) \), thereby showing that all \( re \) sets are the homomorphic images of CSLs.