Sample Question#1. Part a

1. Prove that the following are equivalent

a) $S$ is an infinite recursive (decidable) set.

b) $S$ is the range of a monotonically increasing total recursive function.

Note: $f$ is monotonically increasing means that $\forall x \ f(x+1) > f(x)$.

a) Implies b)

Let $x \in S \iff \chi_S(x)$

Define $f_R(0) = \mu x \chi_S(x)$; $f_R(y+1) = \mu x \ [\chi_S(x) \&\& (x > f_R(y))]$

Clearly, since $S$ is non-empty, it has a least one value and so there exist a smallest value such that $\chi_S(x)$; we will enumerate this as $f_R(0) = \mu x \chi_S(x)$.

Assume we have enumerated the $y$-th value in $S$ as $f_R(y)$. Since $S$ is infinite, there will be values in $S$ greater than $f_R(y)$ and our search $\mu x \ [\chi_S(x) \&\& (x > f_R(y))]$ will find the next largest value for which $\chi_S(x)$. Thus, inductively, we will enumerate the elements of $S$ in increasing order, as desired.
1. Prove that the following are equivalent
a) S is an infinite recursive (decidable) set.
   b) S is the range of a monotonically increasing total recursive function.
      Note: f is monotonically increasing means that \( \forall x \ f(x+1) > f(x) \).
b) Implies a)
Let \( S \) be enumerated by the monotonically increasing algorithm \( f_S \).
Define \( \chi_S \) by
\[
\chi_S(x) = (f_R ((\mu z \ [f_R (z) \geq x]) == x))
\]
Clearly, if \( x \) is enumerated, it must appear before any values greater than it are enumerated and consequently this is a bounded search to find the first element listed that is at least as large as \( x \). If this element is \( x \), then \( x \) is in \( S \), else it is not. The fact that \( f_R \) is monotonically increasing makes \( S \) infinite. The fact that it has a characteristic function makes it decidable.
Sample Question#2

2. Let $A$ and $B$ be re sets. For each of the following, either prove that the set is re, or give a counterexample that results in some known non-re set.

Let $A$ be semi decided by $f_A$ and $B$ by $f_B$

a) $A \cup B$: must be re as it is semi-decided by
   \[ f_{A \cup B}(x) = \exists t \left[ \text{stp}(f_A, x, t) \mathbin{||} \text{stp}(f_B, x, t) \right] \]

b) $A \cap B$: must be re as it is semi-decided by
   \[ f_{A \cap B}(x) = \exists t \left[ \text{stp}(f_A, x, t) \mathbin{\&\&} \text{stp}(f_B, x, t) \right] \]

c) $\neg A$: can be non-re. If $\neg A$ is always re, then all re are recursive as any set that is re and whose complement is re is decidable. However, $A = K$ is a non-rec, re set and so $\neg A$ is not re.
Sample Question#3

3. Present a demonstration that the *even* function is primitive recursive.

\[
even(x) = 1 \text{ if } x \text{ is even} \\
even(x) = 0 \text{ if } x \text{ is odd}
\]

You may assume only that the base functions are prf and that prf’s are closed under a finite number of applications of composition and primitive recursion.

DONE in class.
Sample Question#4

4. Given that the predicate \textbf{STP} and the function \textbf{VALUE} are prf’s, show that we can semi-decide

\[ \{ f \mid \varphi_f \text{ evaluates to 0 for some input} \} \]

This can be shown re by the predicate

\[ \{ f \mid \exists <x,t> \ [ \text{stp}(f,x,t) \ \&\& \ \text{value}(f,x,t) = 0] \} \]
Sample Question#5

5.  Let $S$ be an re (recursively enumerable), non-recursive set, and $T$ be re, non-empty, possibly recursive set. Let $E = \{ z \mid z = x + y, \text{ where } x \in S \text{ and } y \in T \}.$

(a)  Can $E$ be non re?  
No as we can let $S$ and $T$ be semi-decided by $f_S$ and $f_T$, resp., $E$ is then semi-dec. by $f_E(z) = \exists <x,y,t> [\text{stp}(f_S, x, t) \land \text{stp}(f_T, y, t) \land (z = \text{value}(f_S, x, t) + \text{value}(f_T, y, t))]$

(b)  Can $E$ be re non-recursive?  Yes, just let $T = \{0\}$, then $E = S$ which is known to be re, non-rec.

(c)  Can $E$ be recursive?  Yes, let $T = \mathbb{N}$, then $E = \{ x \mid x \geq \text{min} \ (S) \}$ which is a co-finite set and hence rec.
Sample Question#6

6. Assuming TOTAL is undecidable, use reduction to show the undecidability of
Incr = \{ f \mid \forall x \ \varphi_f(x+1) > \varphi_f(x) \}

Let f be arb.

Define \( G_f(x) = \varphi_f(x) - \varphi_f(x) + x \)

\( f \in \text{TOTAL} \) iff \( \forall x \varphi_f(x) \downarrow \) iff \( \forall x \ G_f(x) \downarrow \) iff

\( \forall x \varphi_f(x) - \varphi_f(x) + x = x \) iff \( G_f \in \text{Incr} \)
Sample Question#7

7. Let \( \text{Incr} = \{ f \mid \forall x, \varphi_f(x+1) > \varphi_f(x) \} \).

Let \( \text{TOT} = \{ f \mid \forall x, \varphi_f(x) \downarrow \} \).

Prove that \( \text{Incr} \equiv_m \text{TOT} \). Note Q#6 starts this one.

Let \( f \) be arb.

Define \( G_f(x) = \exists t [\text{stp}(f,x,t) \&\& \text{stp}(f,x+1,t) \&\& (\text{value}(f,x+1,t) > \text{value}(f,x,t))] \)

\( f \in \text{Incr} \) iff \( \forall x \varphi_f(x+1) > \varphi_f(x) \) iff \( \forall x G_f(x) \downarrow \) iff \( G_f \in \text{TOT} \)
8. Let \( \text{Incr} = \{ f \mid \forall x \, \varphi_f(x+1) > \varphi_f(x) \} \). Use Rice’s theorem to show \( \text{Incr} \) is not recursive.

Non-Trivial as
\[
C_0(x) = 0 \notin \text{Incr}; \quad S(x) = x+1 \in \text{Incr}
\]

Let \( f, g \) be arb. Such that \( \forall x \, \varphi_f(x) = \varphi_g(x) \)
\[
f \in \text{Incr} \iff \forall x \, \varphi_f(x+1) > \varphi_f(x) \quad \text{iff} \quad \forall x \, \varphi_g(x+1) > \varphi_g(x) \quad \text{iff} \quad g \in \text{Incr}
\]
9. Let $S$ be a recursive (decidable set), what can we say about the complexity (recursive, re non-recursive, non-re) of $T$, where $T \subseteq S$?

Nothing. Just let $S = \mathbb{N}$, then $T$ could be any subset of $\mathbb{N}$. There are an uncountable number of such subsets and some are clearly in each of the categories above.
Sample Question#10

10. Define the pairing function \(<x,y>\) and its two inverses \(<z>_1\) and \(<z>_2\), where if 
z = \(<x,y>\), then \(x = <z>_1\) and \(y = <z>_2\).

Right out of Notes.
11. Assume $A \leq_m B$ and $B \leq_m C$.
Prove $A \leq_m C$.

Done in class
12. Let $P = \{ f \mid \exists x \ [ \text{STP}(f, x, x) ] \}$. Why does Rice’s theorem not tell us anything about the undecidability of $P$?

This is not an I/O property as we can have implementations of $C_0$ that are efficient and satisfy $P$ and others that do not.