Generally useful information.

- The notation $\mathbf{z = \langle x,y \rangle}$ denotes the pairing function with inverses $\mathbf{x = \langle z \rangle_1}$ and $\mathbf{y = \langle z \rangle_2}$.
- The minimization notation μ **y [P(…,y)]** means the least **y** (starting at **0**) such that **P(…,y)** is true. The bounded minimization (acceptable in primitive recursive functions) notation μ **y (u≤y≤v) [P(…,y)]** means the least **y** (starting at **u** and ending at **v**) such that **P(…,y)** is true. Unlike the text, I find it convenient to define μ **y (u≤y≤v) [P(…,y)]** to be **v+1**, when no **y** satisfies this bounded minimization.
- The tilde symbol, **~**, means the complement. Thus, set **~S** is the set complement of set **S**, and predicate **~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 $\mathbf{y \times P(x)}$, which would evaluate to either **y** (if **P(x)**) or **0** (if **~P(x)**).
- A set **S** is recursive if **S** has a total recursive characteristic function χ_S, such that $\mathbf{x \in S} \Leftrightarrow \chi_S(\mathbf{x})$. Note χ_S is a predicate. Thus, it evaluates to **0** (**false**), if $\mathbf{x \notin S}$.
- 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 $\mathbf{f_S}$.
  2. **S** is the domain of a partial recursive function $\mathbf{g_S}$.
- 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 $\mathbf{\Phi_g(x) = \Phi(x, g) = g(x)}$. Here $\mathbf{\Phi}$ is a universal partially recursive function.
  Moreover, there is a primitive recursive function **STP**, such that
  STP(g, x, t) is **1** (true), just in case **g**, started on **x**, halts in **t** or fewer steps.
  STP(g, x, t) is **0** (false), otherwise.
  Finally, there is another primitive recursive function **VALUE**, such that
  VALUE(g, x, t) is **g(x)**, whenever **STP(g, x, t)**.
  VALUE(g, x, t) is defined but meaningless if **~STP(g, x, t)**.
- The notation $\mathbf{f(x)}\downarrow$ 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 $\mathbf{f(x)}\uparrow$ means **f** diverges when computing with input **x**. In effect, this just means that **x** is **<u>not</u>** 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 $\mathbf{f(x)}\downarrow$. The set of all such pairs, $\mathbf{K_0}$, 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.
- $\mathbf{A \le_m B}$ (**A** many-one reduces to **B**) means that there exists a total recursive function **f** such that $\mathbf{x \in A} \Leftrightarrow \mathbf{f(x) \in B}$. If $\mathbf{A \le_m B}$ and $\mathbf{B \le_m A}$ then we say that $\mathbf{A \equiv_m B}$ (**A** is many-one equivalent to **B**). If the reducing function is 1-1, then we say $\mathbf{A \le_1 B}$ (**A** one-one reduces to **B**) and $\mathbf{A \equiv_1 B}$ (**A** is one-one equivalent to **B**).

COT 6410 Spring 2014 Midterm#1 Name:__________________________________

12 **1**. Choosing from among **(REC) recursive**, **(RE) re non-recursive, (coRE) co-re non-recursive**, **(NRNC) non-re/non-co-re**, categorize each of the sets in a) through d). Justify your answer by showing some minimal quantification of some known recursive predicate.

a.) { f | there is a constant C such that, for every x, f(x) ≤ C, } ____________________

Justification:

b.) { <f,x,c> | f(x) halts in no fewer than c*x+1 steps } ____________________

Justification:

c.) { f | range(f) ⊆ {0,1} // This means range can be {}, {0}, {1} or {0,1} } ____________________

Justification:

d.) { f | range(f) contains at least two elements } ____________________

Justification:

6 **2**. 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 = ~A ____

b.) D = B ∩ ~A ____

c.) D ⊆ C ____

d.) D = A – C ____

6 **3.** Prove that the **Uniform Halting Problem** (the set **TOTAL**) is not recursive enumerable within any formal model of computation. (Hint: A diagonalization proof is required here.)

5 **4.** Using many-one reduction from the known non-recursive set **HasADouble**, where **HasADouble = { f | $\exists x \varphi_f(x)=2*x$ }**, show that **IsDouble** is non-recursive, where **IsDouble = { f | $\forall x \varphi_f(x)=2*x$ }**
Just giving a construction is not sufficient; you must also explain why it satisfies the desired properties of the reduction.

5. Define **NullDomain** as **ND = { f | for all x $\varphi_f(x)\uparrow$** }.

3 **a.)** Show some minimal quantification of some known recursive predicate that provides an upper bound for the complexity of this set. (Hint: Look at **c.)** and **d.)** to get a clue as to what this must be.)

5 **b.)** Use Rice's Theorem to prove that **ND** is undecidable.

(Problem Continued on next page)

5 **c.)** Show that **NotHalt $\leq_m$ ND**, where **NotHalt = { <f,x> | $\varphi_f(x)\uparrow$ }**. Justify your construction.

5 **d.)** Show that **ND $\leq_m$ NotHalt**. Justify your construction.

3 **e.)** From **(a.)** through **(d.)** what can you conclude about the complexity of **ND** (choose from **REC, RE, RE-MANY-ONE-COMPLETE, CO-RE, CO-RE-MANY-ONE -COMPLETE, NON-RE/NON-CO-RE**)? Briefly justify your conclusion, stating what each of **(a)**, **(b)**, **(c)** and **(d)** show.

6 **6.** Rice's Theorem has a strong and two weak forms. Given the problem of determining membership in
IsDouble = { f | $\forall x \varphi_f(x)=2*x$ }
Show how the strong form can prove this undecidable, but the weak forms cannot. Be sure to cover all conditions that must apply, indicating what is common between the three forms and what is not.

6 **7.** Let **S** be an arbitrary infinite re set. Furthermore, let **S** be the range of some total recursive function $\mathbf{f_s}$. Show that **S** has an infinite recursive subset enumerated by some monotonically increasing total recursive function $\mathbf{g_s}$. You must give an explicit definition of $\mathbf{g_s}$ that you form from $\mathbf{f_s}$. Justify that your $\mathbf{g_s}$ enumerates a subset of S and that it is monotonically increasing. The fact that $\mathbf{g_s}$ is a monotonically increasing function is sufficient to show the subset it enumerates is infinite and recursive, so you do not have to show those properties.