COT 6410 Spring 2014 Final Exam E1 Sample Name:__________________

- The notation $\mathbf{z = <x,y>}$ denotes the pairing function with inverses $\mathbf{x = <z>_1}$ and $\mathbf{y = <z>_2}$.
- 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. I 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 the 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 **y × 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 $\mathbf{x \in S \Leftrightarrow \chi_S(x)}$. Note χ_S is a total 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** (we tend to overload the index as the function name), such that $\mathbf{\Phi_g(x) = \Phi(x, g) = g(x)}$. Here **Φ** 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 **f(x)↓** means that **f** converges when computing with input **x** (**x ∈ Dom(f)**). The notation **f(x)↑** means **f** diverges when computing with input **x** (**x ∉ 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)↓**. The set of all such pairs, $\mathbf{K_0}$, is a classic re non-recursive set. $\mathbf{K_0}$ is also known as $\mathbf{L_u}$, the universal language. The related set, **K**, is the set of all effective procedures **f** such that **f(f)↓** or more precisely $\mathbf{\Phi_f(f)}$.
- 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, **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 $\mathbf{G_f}$, another unary function, by
  $\mathbf{G_f(0) = f(0); G_f(y+1) = G_f(y) + f(y+1)}$
  This would get $\mathbf{G_f(x)}$ as the sum of the values of **f(0)+f(1)+…+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 $\mathbf{<x_1,x_2,\ldots,x_n>, <y_1,y_2,\ldots,y_n>}$. The question is whether or not there exists a sequence, $\mathbf{i_1,i_2,\ldots,i_k}$, such that $\mathbf{1 \le i_j \le n}$, $\mathbf{1 \le j \le k}$, and $x_{i_1}x_{i_2}\cdots x_{i_k} = y_{i_1}y_{i_2}\cdots y_{i_k}$

- When I ask you to show one set of indices, **A**, is many-one reducible to another, **B**, denoted **A ≤m B,** you must demonstrate a total computable function **f**, such that **x ∈ A ⇔ f(x) ∈ B**. The stronger relationship is that **A** and **B** are many-one equivalent, **A ≡m B**, requires that you show **A ≤m B** and **B ≤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 ≤1 B**.

COT 6410 Spring2014 Final Exam Sample E1 Questions

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.) $\mathbf{D = \sim C}$ ____

b.) $\mathbf{D \subseteq (A \cup C)}$ ____

c.) $\mathbf{D = \sim B}$ ____

d.) $\mathbf{D = B - A}$ ____

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

Problem / Language Class	**Regular**	**Context Free**	**Context Sensitive**
$\mathbf{L = \Sigma^* \ ?}$			
$\mathbf{L = \phi \ ?}$			
$\mathbf{L = L^2 \ ?}$			
$\mathbf{x \in L^2}$**, for arbitrary x ?**			

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 $\mathbf{G_A}$ and $\mathbf{G_B}$ based on some instance $\mathbf{P = \langle\langle x_1,x_2,\ldots,x_n\rangle, \langle y_1,y_2,\ldots,y_n\rangle\rangle}$ of **PCP**, such that $\mathbf{L(G_A) \cap L(G_B) \neq \phi}$ iff **P** has a solution. Assume that **P** is over the alphabet $\mathbf{\Sigma}$. 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 **CONSTANT** = $\{ f \mid \exists K\ \forall y\ [\ \varphi_f(y) = K\ ]\ \}$. Use Rice's Theorem to show that **CONSTANT** is not recursive. Hint: There are two properties that must be demonstrated.

5. Show that **CONSTANT** $\equiv_m$ **TOT**, where **TOT** = $\{ f \mid \forall y\ \varphi_f(y)\downarrow\ \}$.

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.
AT-LEAST-LINEAR = { f | $\forall$y φ_f(y) converges in no fewer than y steps }.

7. The trace language of a computational device like a Turing Machine is a language of the form
Trace = { $C_1\#C_2\# \ldots C_n\#$ | $C_i \Rightarrow C_{i+1}$, $1 \le i < n$ }
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, Σ, S, P)** is an arbitrary Phrase Structured Grammar, with **N** its set of non-terminals, **Σ** 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', Σ', 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.