**Generally useful information.**

* The notation **z =** **<x,y>** denotes the pairing function with inverses **x =** **<z>1** and **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. 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 **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 **χS**, such that **x ∈ S ⇔ χS(x)**. Note **χS** is a predicate. Thus, it evaluates to **0** (**false**), if **x ∉ 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 **fS**.
2. **S** is the domain of a partial recursive function **gS**.
* 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 **Φg(x) = Φ(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**, 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)↑** 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)↓**. The set of all such pairs, **K0**, 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.
* **A ≤m B** (**A** many-one reduces to **B**) means that there exists a total recursive function **f** such that
**x ∈ A ⇔ f(x) ∈ B**. If **A ≤m B** and **B ≤m A** then we say that **A ≡m B** (**A** is many-one equivalent to **B)**. If the reducing function is 1-1, then we say **A ≤1 B** (**A** one-one reduces to **B**) and **A ≡1 B** (**A** is one-one equivalent to **B**).

**COT 6410 Spring 2015 Sample Midterm#1 Name:**

 **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 | domain(f) is finite }**

 **Justification:**

**b.) { f | domain(f) is empty }**

 **Justification:**

**c.) { <f,x> | f(x) converges in at most 20 steps }**

**Justification:**

**d.) { f | domain(f) converges in at most 20 steps for some input x }**

 **Justification:**

 **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 = ~C**

**b.) D ⊆ A ∪ C**

**c.) D = ~B**

**d.) D = B − A**

 **3.** Prove that the **Halting Problem** (the set **HALT** **=** **K0 = Lu**) is not recursive (decidable) within any formal model of computation. (Hint: A diagonalization proof is required here.)

***Look at notes.***

 **4.** Using reduction from the known undecidable **HasZero, HZ = { f | ∃x f(x) = 0 }**, show the non-recursiveness (undecidability) of the problem to decide if an arbitrary partial recursive function **g** has the property **IsZero, Z = { f | ∀x f(x) = 0 }**. Hint: there is a very simple construction that uses **STP** to do this. **Just giving that construction is not sufficient; you must also explain why it satisfies the desired properties of the reduction**.

 **5.** Define **RANGE\_ALL = ( f | range(f) = ℵ }**.

 **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.)

 **b.)** Use Rice’s Theorem to prove that **RANGE\_ALL** is undecidable.

 **c.)** Show that **TOTAL ≤m RANGE\_ALL**, where **TOTAL = { f | ∀y ϕf(y)↓ }**.

 **d.)** Show that **RANGE\_ALL ≤m TOTAL**.

 **e.)** From **a.)** through **d.)** what can you conclude about the complexity of **RANGE\_ALL**?

 **6.** This is a simple question concerning Rice’s Theorem.

**a.)** State the strong form of Rice’s Theorem. Be sure to cover all conditions for it to apply.

 **b.)** Describe a set of partial recursive functions whose membership cannot be shown undecidable through Rice’s Theorem. What condition is violated by your example?

 **7.** Using the definition that **S** is recursively enumerable iff **S** is either empty or the range of some algorithm **fS** (total recursive function), prove that if both **S** and its complement **~S** are recursively enumerable then **S** is decidable. To get full credit, you must show the characteristic function for **S**, **χS**, in all cases. Be careful to handle the extreme cases (there are two of them). Hint: This is not an empty suggestion.