**COT 6410 Spring 2014 Midterm#1 Name: KEY**

***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, }**  ***NRNC***

**Justification: *∃C∀x∃t [STP(f,x,t)&&VALUE(f,x,t)≤C]***

**If interpret as not requiring convergence then *∃C∀<x,t> [STP(f,x,t)⇒VALUE(f,x,t)≤C]***

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

**Justification: ~STP(f,x,c\*x)**

**c.) { f | range(f) ⊆ {0,1} // This means range can be {}, {0}, {1} or {0,1} }**  ***co-RE***

**Justification: *∀<x,t> [STP(f,x,t)⇒VALUE(f,x,t)≤1]***

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

**Justification: *∃<x,y,t> [STP(f,x,t)&STP(f,y,t)&(VALUE(f,x,t)≠VALUE(f,y,t))]***

***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 *REC***

**b.) D = B ∩ ~A *REC, RE***

**c.) D ⊆ C *REC, RE, NR***

**d.) D = A − C *REC, RE, NR***

***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.)

***Assume TOTAL is re. As it is a non-empty set, e.g., the index of the function C0 is in TOTAL, then there must exist a total recursive function that enumerates TOTAL. Call this function A. Thus, the algorithms have indices A(0), A(1), …We also know that any complete model of computation must have a universal function. Call it ϕ. Here ϕ (f,x) is just ϕ f (x), that is, the function whose index is f evaluated at x.***

***Define a new function D(x) = ϕ(A(x),x) + 1. Clearly D is an algorithm as it just evaluates the x-th algorithm on the input x and then adds 1. As D is an algorithm, its index is in TOTAL and so is the d-th algorithm enumerated by A, for some natural number d.***

***Consider D(d). D(d) = ϕ(A(d),d) + 1 by D’s definition. But, then  
D(d) = ϕ(A(d),d) + 1 = ϕA(d)(d) + 1 = D(d) + 1***

***The above is clearly a contradiction since D is an algorithm.   
This means that TOTAL cannot be re.***

***5*****4.** Using many-one redu#3ction from the known non-recursive set **HasADouble**, where **HasADouble = { f | ∃xϕf (x)=2\*x }**, show that **IsDouble** is non-recursive,   
where **IsDouble = { f | ∀xϕf (x)=2\*x }**  
Just giving a construction is not sufficient; you must also explain why it satisfies the desired properties of the reduction.

***Let f be an arbitrary natural number.***

***Define Gf,x(y) = 2 \* y \* ∃<x,t> [STP(f,x,t)&VALUE(f,x,t) = 2\*x]***

***Thus, f ∈ HasADouble iff ∃xϕf (x)=2\*x iff* *∃<x,t> [STP(f,x,t)&VALUE(f,x,t) = 2\*x]   
iff ∀yGf,x(y) = 2 \* y iff Gf,x ∈ IsDouble.***

***This show HasADouble ≤m IsDouble and so a solution to IsDouble implies one to HasADouble, which is known to be non-recursive. Thus, IsDouble must also be non-recursive.***

**5.** Define **NullDomain** as **ND = { f | for all x ϕf(x)↑ }**.

***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.)

***∀<x,t> [~STP(f,x,t)] shows ND is at worst co-RE***

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

***Clearly ↑ ∈ ND, where ↑ diverges for all input. However, C0 ∉ ND as it converges everything so has a non-empty domain. Thus, ND is non-trivial.***

***Let f, g be arbitrary unique natural numbers such that dom(f) = dom(g).***

***f ∈ ND ⇔ dom(f) = ∅ ⇔ dom(g) = ∅ iff g ∈ ND***

***This satisfies the second property of one of the weak forms of Rice’s Theorem and so ND is undecidable.***

***5*** **c.)** Show that **NotHalt ≤m ND**, where **NotHalt = { <f,x> | ϕf(x)↑ }**. Justify your construction.

***Let f, x be an arbitrary pair of natural numbers. Define Gf,x(y) = f(x).***

***We can see that Dom(Gf,x) = ∅ iff ϕf(x)↑***

***Thus, <f,x> ∈ NotHalt iff ϕf(x)↑ iff ∀yGf,x(y)↑ iff Gf,x ∈ ND***

***5*** **d.)** Show that **ND ≤m NotHalt**. Justify your construction.

***Let f be an arbitrary natural number. Define Gf(y) = ∃<x,t> [STP(f,x,t)].***

***f ∈ ND iff ∀<x,t> [~STP(f,x,t)] iff ~∃<x,t> [STP(f,x,t)] iff ∀yGf(y)↑ iff < Gf, 0> ∈ NotHalt***

***Here, we note that Gf is either the constant 1 (converges everywhere) or diverges everywhere. Hence we can choose any natural number as the argument to Gf when checking membership in NotHalt.***

***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.

***ND is co-RE Many-One Complete***

***a) Shows ND is co-RE; b) Shoes ND is non-recursive; c) Shows ND is at least as hard as NotHalt, which is the complement of Halt, a Complete RE m-1 problem; d) adds nothing.***

***6* 6.** Rice’s Theorem has a strong and two weak forms. Given the problem of determining membership in  **IsDouble = { f | ∀xϕf (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.

***Clearly, d(x) = 2x ∈ IsDouble However, C0 ∉ IsDouble. Thus, IsDouble is non-trivial.***

***Strong Form:***

***Let f, g be arbitrary unique natural numbers such that ∀xϕf (x) = ϕg (x)***

***f ∈ IsDouble ⇔ ∀xϕf (x)= 2\*x ⇔ ∀xϕg (x)= 2\*x iff g ∈ IsDouble***

***This satisfies the second property of one of the strong forms of Rice’s Theorem and so IsDouble is undecidable.***

***Weak Form #1 (Domains)***

***Let f, g be arbitrary unique natural numbers such that dom(ϕf) = dom(ϕg)***

***Consider ϕf(x) = 2x and ϕf(x) = 2x-2. Both have domains of all natural numbers and yet one is in IsDouble and the other is not, so this form does not give us the desired result.***

***Weak Form #2 (Ranges)***

***Let f, g be arbitrary unique natural numbers such that range(ϕf) = range(ϕg)***

***Consider ϕf(x) = 2x and ϕf(x) = 2x-2. Both have ranges of all even natural numbers and yet one is in IsDouble and the other is not, so this form does not give us the desired result.***

***6* 7.** Let **S** be an arbitrary infinite re set. Furthermore, let **S** be the range of some total recursive function **fs**. Show that **S** has an infinite recursive subset enumerated by some monotonically increasing total recursive function **gs**. You must give an explicit definition of **gs** that you form from **fs**. Justify that your **gs** enumerates a subset of S and that it is monotonically increasing. The fact that **gs** 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.

***Define gS using primitive recursion as follows:***

***gS(0) = fS(0)***

***gS(x) = fS(μ z [ fS(z) > gS(x) ])***

***First, since S is infinite, it has a first element fS(0) and it has no largest number.***

***The first property gives us the basis for gS and the second guarantees that the search,   
μ z [ fS(z) > gS(x) ] will always succeed in finding a value in the range of fS that is greater than all previous values enumerated by gS.***

***Clearly the range of gS is a subset of that of fS and gS is monotonically increasing. Thus, the range of gS is an infinite recursive subset of S.***