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 χ_S, such that **x ∈ S ⇔** χ_S**(x)**. Note χ_S 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 f_S.
  2. **S** is the domain of a partial recursive function 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 $\Phi_g(x) = \Phi(x, g) = g(x)$. 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**, 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, 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.
- $A \le_m B$ (**A** many-one reduces to **B**) means that there exists a total recursive function **f** such that **x ∈ A ⇔ f(x) ∈ B**. If $A \le_m B$ and $B \le_m A$ then we say that $A \equiv_m B$ (**A** is many-one equivalent to **B**). If the reducing function is 1-1, then we say $A \le_1 B$ (**A** one-one reduces to **B**) and $A \equiv_1 B$ (**A** is one-one equivalent to **B**).

COT 6410 Spring 2015 Sample Midterm#1 Name: KEY

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 } NRNC

   Justification: ∃x ∀y≥x ∀t ~STP(f, y, t)

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

   Justification: ∀x ∀t ~STP(f, x, t)

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

   Justification: STP(f, x, 20)

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

   Justification: ∃x STP(f, x, 20)

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 **RE, NR**

   b.) D ⊆ A ∪ C **REC, RE, NR**

   c.) D = ~B **NR**

   d.) D = B – A **REC, RE**

3. Prove that the **Halting Problem** (the set **HALT = K_0 = L_u**) 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**.

   HZ = { f | ∃x ∃t [STP(f, x, t) & VALUE(f, x, t) == 0] }
   Let f be the index of an arbitrary effective procedure.
   Define $g_f(y)$ = 1 - ∃x ∃t [STP(f, x, t) & VALUE(f, x, t) == 0]
   If ∃x f(x) = 0, we will find the x and the run-time t, and so we will return 0 (1 – 1)
   If ∀x f(x) ≠ 0, then we will diverge in the search process and never return a value.
   Thus, f ∈ HZ iff g_f ∈ Z.

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.)

$\forall x\ \exists \langle y,t\rangle [STP(f,y,t)\ \&\&\ Value(f,y,t)=x]$

b.) Use Rice's Theorem to prove that **RANGE_ALL** is undecidable.
This is non-trivial as $I(x) = x \in$ RANGE_ALL and $C_0(x) = 0 \notin$ RANGE_ALL
Let f,g be such that $\forall x\ \varphi_f(x) = \varphi_g(x)$.
f∈ RANGE_ALL ⇔ range(f) = ℵ
⇔ range(g) = ℵ since g outputs the same value as f for any input
⇔ g ∈ RANGE_ALL
Since the property is non-trivial and is an I/O property, Rice's Theorem says it is undecidable.

c.) Show that **TOTAL $\leq_m$ RANGE_ALL**, where **TOTAL = { f | $\forall y\ \varphi_f(y)\downarrow$** }.

Let f be the index of an arbitrary effective procedure φ_f. Define g such that g(f), denoted g_f, is the index of the function φ_{g_f} defined by $\varphi_{g_f}(x) = \varphi_f(x) - \varphi_f(x)+x$.

$f \in$ TOTAL $\Leftrightarrow \forall x\ \varphi_f(x)\downarrow \Leftrightarrow \forall x\ \varphi_{g_f}(x) = x \Rightarrow \forall x\ x \in range(g_f) \Rightarrow g_f \in$ RANGE_ALL

$f \notin$ TOTAL $\Leftrightarrow \exists x\ \varphi_f(x)\uparrow \Leftrightarrow \exists x\ \varphi_{g_f}(x)\uparrow \Rightarrow \exists x\ x \notin range(g_f) \Rightarrow g_f \notin$ RANGE_ALL

This shows that TOTAL $\leq_m$ RANGE_ALL, as was desired.

d.) Show that **RANGE_ALL $\leq_m$ TOTAL**.

Let f be the index of an arbitrary effective procedure φ_f. Define g such that g(f), denoted g_f, is the index of the function φ_{g_f} defined by $\varphi_{g_f}(x) = \exists \langle y,t\rangle\ [STP(f,y,t)\ \&\ Value(f,y,t)=x]$.

$f \in$ RANGE_ALL $\Leftrightarrow \forall x\ \exists \langle y,t\rangle\ [STP(f,y,t)\ \&\&\ Value(f,y,t)=x] \Leftrightarrow \forall x\ \varphi_{g_f}(x)\downarrow \Leftrightarrow g_f \in$ TOTAL

This shows that RANGE_ALL $\leq_m$ TOTAL, as was desired.

e.) From **a.)** through **d.)** what can you conclude about the complexity of **RANGE_ALL**?
a) shows that RANGE_ALL is no more complex than others that must use the alternating qualifiers ∀∃. b) shows the problem is non-recursive. c) and d) combine to show that the problem is in fact of equal complexity with the non-re problem TOTAL, so the result in a) was optimal.

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.

Let P be a property of indices of partial recursive function such that the set S_P = { f | f has property P } has the following two restrictions

(1) S_P is non-trivial. This means that S_P is neither empty nor is it the set of all indices.

(2) P is an I/O behavior. That is, if f and g are two partial recursive functions whose I/O behaviors are indistinguishable, $\forall x\ f(x)=g(x)$, then either both of f and g have property P or neither has property P.

Then P is undecidable.

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?

There are many possibilities here. For example { f | $\exists x$ ~STP(f,x,x) } is not an I/O property and { f | $\exists x\ f(x) \neq f(x)$ } is trivial (empty).

7. Using the definition that **S** is recursively enumerable iff **S** is either empty or the range of some algorithm $\mathbf{f_S}$ (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**, χ_S, in all cases. Be careful to handle the extreme cases (there are two of them). Hint: This is not an empty suggestion.

Let S = ϕ then ~S = $\aleph$. Both are re and $\forall x\ \chi_S(x) = 0$ is S's characteristic function.

Let S = $\aleph$ then ~S = ϕ. Both are re and $\forall x\ \chi_S(x) = 1$ is S's characteristic function.

Assume then that S $\neq \phi$ and S $\neq \aleph$ then each of S and ~S is enumerated by some total recursive function. Let S be enumerated by f_S and ~S by $f_{\sim S}$. Define

$\chi_S(x) = f_S(\ \mu y\ [f_S(y)==x\ ||\ f_{\sim S}(y)==x]\ ) == x$.

Moreover, the minimization, while conceptually unbounded, always converges because both f_S and by $f_{\sim S}$ are algorithms.

Further, x must be in the range of one and only one of f_S or $f_{\sim S}$. Thus, $\exists y\ f_S(y) == x$ or $\exists y\ f_{\sim S}(y) == x$.

The min operator (μy) finds the smallest such y and the predicate

$f_S(\ \mu y\ [f_S(y)==x\ ||\ f_{\sim S}(y)==x]\ ) == x$ checks that x is in the range of f_S.

If it is, then $\chi_S(x) = 1$ else $\chi_S(x) = 0$, as desired.