1. (15 points) Give examples of languages satisfying the following conditions or state that none exist. Do not give proofs.
   (a) A language $A$ such that $A$ is regular but $A^*$ is not regular.
   (b) A language $A$ such that $A$ is not regular but $A^*$ is regular.
   (c) A recursive language with a non-recursive subset.

2. (20 points)
   (a) Let $\Sigma = \{0, 1, \#\}$. Let $A = \{x\#y\#z \mid x, y, z \in \{0,1\}^* \text{ and either } x = z^R \text{ or } y = z^R\}$. Give a context free grammar generating the language $A$. (You do not have to prove or justify that it works.)
   (b) Is the grammar you gave in part (a) ambiguous? Why or why not?
   (c) Give a context-free grammar generating the language described by $(bb \cup a)^* (aa \cup b)^*$. 

3. (20 points) Show that a subset of $L = \{a^n b^n \mid n \geq 0\}$ is regular if and only if it is finite.

4. (20 points) Let $EQ_{TM} = \{<M, N> \mid M, N \text{ are Turing machines such that } L(M) = L(N)\}$. Prove that this language is undecidable.

5. (25 points) A deterministic (or nondeterministic) counter automaton, DCA (NCA), is a deterministic (or nondeterministic) pushdown automaton which has the ability to write only one type of symbol on the stack. The stack, as usual, is initialized with a special bottom of stack symbol. Thus the automaton can use the stack as a counter which can be incremented or decremented, or tested for equality to zero. The automaton accepts by entering a final state at the end of the input string. A deterministic (or nondeterministic) counter language, DCL (NCL) is a language accepted by some deterministic (or nondeterministic) counter automaton.

Answer the following parts, giving short reasons for your answers:
   (a) Is the class of deterministic counter languages closed under complementation?
   (b) Is the class of nondeterministic counter languages closed under union?
   (c) Is the class of deterministic counter languages closed under intersection?
   (d) Let $E_{dca} = \{<M> \mid M \text{ is a deterministic counter automaton accepting the empty language}\}$. Is $E_{dca}$ recursive?
Solutions to Sample Exam #1

1. (a) None exists. Regular languages are closed under the * operation.
   (b) \( A = \{ w | \text{ } w = w^R, w \in \Sigma^* \} \). (\( A \) is the set of all palindromes.) Note that \( A^* = \Sigma^* \).
   (c) \( \Sigma^* \). This language is clearly recursive, but \( L_u \) (i.e., \( A_{TM} = \text{ Universal language} \) \( \subseteq \Sigma^* \) is not.

2. (a) \( S \rightarrow A | E\#B \)
   \[ \begin{align*}
   A & \rightarrow 0A0 | 1A1 | \#E\# \quad \{ \text{generates } x\#y\#x^R \} \\
   B & \rightarrow 0B0 | 1B1 | \# \quad \{ \text{generates } y\#y^R \} \\
   E & \rightarrow 0E | 1E | \varepsilon \quad \{ \text{generates all strings} \}
   \end{align*} \]
   (b) Yes. If \( x = y \) then we generate \( x\#y\#x^R \) or \( x\#y\#y^R \). These can generate the same string with different parse trees.
   (c) \( S \rightarrow AB \)
   \[ \begin{align*}
   A & \rightarrow CA | \varepsilon \quad \{ A \text{ generates } (bb \cup a)^* \} \\
   B & \rightarrow DB | \varepsilon \quad \{ B \text{ generates } (aa \cup b)^* \} \\
   C & \rightarrow bb | a \quad \{ C \text{ generates } (bb \cup a) \} \\
   D & \rightarrow aa | b \quad \{ D \text{ generates } (aa \cup b) \}
   \end{align*} \]

3. The (\( \Rightarrow \)) direction is trivial, since any finite set is regular.
   (\( \Leftarrow \)): We will show that any infinite subset of \( L \) is non-regular, thus implying that any regular subset of \( L \) must be finite. Let \( L' \) be some infinite subset of \( L \). Let \( n \) be the pumping length constant of the pumping lemma. Since \( L' \) is infinite, \( \exists z = a^n b^n \in L' \) such that \( m > n \). Now for all \( u, v, w \) such that \( z = uvw \), where \( |uv| \leq n \) and \( |v| \geq 1 \), \( uv^0 w \notin L' \), since the number of \( a \)'s is smaller than the number of \( b \)'s. So, \( L' \) is not regular.

4. Suppose \( EQ_{TM} \) is decided by \( S \). We build \( T \) to decide \( L_u \) (i.e., \( A_{TM} \)) as follows:
   “On input \( <M, w> \):
   Construct \( M' = \text{ “On input } x, \)
   If \( x = w \) then accept
   Else, run \( M \) on \( x' \)"

   Run \( S \) on \( <M, M'> \).”

   Note that \( L(M) = L(M') \Leftrightarrow M \text{ accepts } w \Leftrightarrow <M, w> \in L_u \) (i.e., \( A_{TM} \)).
   So \( T \) decides \( L_u \) (i.e., \( A_{TM} \)), which is a contradiction. Thus, \( EQ_{TM} \) is undecidable.

5. (a) Yes. Given \( M \), a DCA accepting \( L \), we can build a DCA \( M' \) accepting \( \overline{L} \) by exchanging the final and non-final states of \( M \).
   (b) Yes. If NCA \( M_1 \) accepts \( L_1 \) and \( M_2 \) accepts \( L_2 \), then NCA \( M \) accepts \( L_1 \cup L_2 \) by initially guessing which of \( M_1 \) or \( M_2 \) to simulate.
   (c) No. The class of DCLs is clearly a subset of CFLs. So any \( L \) which is not a CFL is also not a DCL. Note that \( L_1 = 0^n1^n2^n \) and \( L_2 = 0^*1^n2^n \) can both be accepted by DCAs. But \( L_1 \cap L_2 = 0^n1^n2^n \) is not context free, and thus not a DCL.
   (d) Yes. To decide \( Edca \), view the input \( M \) as a PDA and convert it to a CFG \( G_M \). Now run TM \( E_{cfg} \) (discussed in class) on \( <G_m> \). Accept iff this TM accepts.