MA/CSSE 474 Exam 3 Winter 2009-10 Name_____**Solution**_____

1. (28 points) For each of the following statements, circle T or F to indicate whether it is *True* or *False*.
If it is sometimes False, you should choose False.
You do not have to give proofs or counterexamples.
For each part, you get 2 points for circling IDK (I don't know), 4 for circling the correct answer, and 0 for circling the incorrect answer or leaving it blank. Reason: When you don't know something, knowing that you don't know counts for something. In order o be lazy in writing this solution, I am using Theorem numbers from the textbook. Of course I did not expect you to know them by number.

   a) T **F** IDK If R is regular and R∩L is context-free, then L is context-free
   Let L be *any* non-context-free language, and R = ∅. Then R∩L = ∅, which is CF
   b) **T** F IDK If R is regular and R∩L is not context-free, then L is not context-free
   This is the contrapositive of Theorem 13.7
   c) T **F** IDK The complement of a context-free language cannot be context-free.
   ∅ is context-free, and so is its complement, Σ*. Note that this does not contradict Theorem 13.6
   d) **T** F IDK Every context-free language is decidable.
   This is Theorem 14.1
   e) T **F** IDK Let L be such that, for each $w \in L$, there exists some DFSM that accepts w. Then L must be regular.
   See the solution of Exam 2.
   f) T **F** IDK If L^+ is context-free, then L must be context-free.
   g) Let L = $\{a^p : p \text{ is a prime integer}\}$. Not context-free. But L^+ is aaa*m which is regular and thus context-free.
   T F IDK If L is context-free, then L^+ must be context-free.
   CFLs are closed under concatenation and Kleene star. $L^+ = LL^*$

2. (32 points) For each of the following statements, circle
R if the language is regular,
CF-R if it is context-free but not regular,
NCF if it is not context-free
IDK if you don't know. Scoring: Correct answer - 4, IDK - 2, incorrect answer - 0.

   a) R **CF-R** NCF IDK $WW^R = \{ww^R : w \in \{a,b\}^*\}$.
   Example 8.11 shows it is not regular by pumping $a^k b^k b^k a^k$. Generated by S→aSa | bSb | ε.
   b) R **CF-R** NCF IDK {u#v^R : u and v are binary encodings (no leading zeroes) of positive integers, where v=2u}
   Context-free not regular. The key is that, if $v = 2u$, then $\langle v \rangle = \langle u \rangle 0$. So every string in L has the form $u\#0u^R$. The following context-free grammar generates L:

   $S \rightarrow 1\ T\ 1$ /* u must start with 1 since no leading 0's.
   $T \rightarrow 0\ T\ 0 \mid 1\ T\ 1 \mid \#\ 0$

   Proof not regular is by pumping. Let $w = 1^k\#01^k$. Then y is 1^p, for some nonzero p, and it must occur in the initial 1 region. Pump in once. The resulting string is $1^{k+p}\#01^k$. But it is not true that $1^k0 = 2\cdot(1^{k+p})$. So this string is not in L.

c) **R** CF-R NCF IDK $\{u\#v : u, v \in \{a, b\}^*$ and $\exists x \in \{a, b\}^*$ (x is a substring of u and x^R is a substring of v)

The key is that x can be equal to ε. So, letting x be ε, this expression can generate exactly the language $(a \cup b)^* \# (a \cup b)^*$. If x takes on any other value, all we get is an alternative derivation for some string that can be generated just from $(a \cup b)^* \# (a \cup b)^*$. So $L = (a \cup b)^* \# (a \cup b)^*$, which is regular.

d) R CF-R **NCF** IDK $\{w = xyz : x \in 0^*, y \in 1^*, z \in 0^*, |x| = |z|$ and $|y| = 2 \cdot |z|\}$

Not context-free. Let $w = 0^k 1^{2k} 0^k$.

1 | 2 | 3

If either v or y from the pumping theorem crosses regions, pump in once. The resulting string will violate the form constraint and so not be in L. We now consider the other ways in which v and y could occur:

(1, 1), (1, 2), (2, 3), (3, 3): Pump in once. The lengths of the x and z regions will no longer be equal because one changed and the other didn't.

(2, 2): Pump out. Since the length of the y region changed and the length of the z region didn't, it will no longer be true that $|y| = 2 \cdot |z|$.

(1, 3): Not possible since $|vxy| \leq k$.

e) **R** CF-R NCF IDK L(G) where G is $S \rightarrow TSb \mid Tb$, $T \rightarrow Ta \mid \varepsilon$.

Regular (even though this particular grammar is not regular). L(G) = a*b*.

f) R **CF-R** NCF IDK $\neg L$, where $L = \{wcw^R : w \in \{a, b\}^*\}$.

L is deterministic context-free. (It can be accepted by a straightforward deterministic PDA that pushes until it gets to the c, then pops matching characters (see Example 12.3.). The deterministic context-free languages are closed under complement.

If $\neg L$ were regular, then L would also be regular since the regular languages are closed under complement. But we show that it is not by pumping. Let $w = a^k c a^k$. Then y is a^p, for some nonzero p, and it must occur in the initial a region. Pump in once. The resulting string is $a^{k+p} c a^k$. It is not in L because there are $k+p$ a's before the c reading from the left but only k a's before the c, reading from the right.

g) **R** CF-R NCF IDK $\{a^i b^j c^k : i, j, k \geq 0$ and $2i + 3j \equiv_3 k\}$ [recall that $\equiv_3$ means "congruent mod 3"]

We can build an FSM or a regular expression for L by dividing its strings into three groups, one for each of the equivalence classes of $\equiv_3$. Doing this, we get the regular expression:

$$(aaa)^* b^* (ccc)^* \cup a (aaa)^* b^* cc (ccc)^* \cup aa (aaa)^* b^* c (ccc)^*$$

Notice that the number of b's is not important. For any j, $3j \equiv_3 0$.

h) R CF-R **NCF** IDK $\{w \in \{a, b, c\}^*$: every a has a matching b and a matching c somewhere in w, and no b or c is considered to match more than one a$\}$

Not context-free. If L were context-free, then $L_1 = L \cap a^*b^*c^*$ would also be context-free. But we show that it is not by pumping. Let $w = a^k b^k c^k$.

1 | 2 | 3

We break w into regions as shown above. If either v or y crosses numbered regions, pump in once. The resulting string will not be in L_1 because it will violate the form constraint. We now consider the other ways in which v and y could occur:

(1, 1): Pump in once. The number of a's went up, but the number of b's didn't so there is no longer a matching b for every a.

(2, 2): Pump out once. The number of b's went down, but the number of a's didn't so there is no longer a matching b for every a.

(3, 3): Pump out once. The number of c's went down, but the number of a's didn't so there is no longer a matching c for every a.

(1, 2): Pump in once. The number of a's went up, but the number of c's didn't so there is no longer a matching c for every a.

(2, 3): Pump out once. The number of c's went down, but the number of a's didn't so there is no longer a matching c for every a.

(1, 3): Not possible since $|vxy| \leq k$.

3. (10 points) Choose a language from problem 2 that is not context-free, and prove that it is not CF. You must use the Pumping Theorem in your proof. [**Hint** (if you want it): For 3 points (plus I will immediately grade that part of problem 2), I will tell you which part of #2 I think is the easiest one to use for this problem. And you will know that the language in that problem is not CF.]

Could do either (d) or (h). Answers are above. If students choose one that is in fact CF, 0 points

4. (10 points) Design a Turing Machine that computes n % m (i.e., the remainder when integer n is divided by integer m) in unary. If the input is $1^n,1^m$ (the comma is part of the input string), the output should be $1^{n\%m}$. Your description may include a transition diagram, one or more of our macro diagrams, and/or a clear English description of your machine.

The burden is on you to convince me that your machine works.

Use a 2-tape machine. Tape 1 alphabet { 1, #, □, ,} (last symbol is comma). Tape 2 alphabet { 1, □ }

Copy m to the second tape, while erasing it and the comma from the first tape.

Move left to the first non-blank symbol on each tape.
Repeat
  Move right on both tapes. As long as there is a 1 on both tapes, replace the 1 on the Tape 1 with a #.
  If a blank is encountered on Tape 1 before Tape 2:
    Move left on both tapes
    While Tape 1 symbol is not blank:
      Write 1 in place of each # on Tape 1 and blank in place of each 1 on Tape 2
      Move left on both tapes
    Halt
If a blank is encountered on Tape 2 before (or at the same time as) **a** blank on Tape 1
    Move left on both tapes
    While Tape 1 symbol is not blank:
      Write a blank in place of each # on Tape 1, and leave tape 2 unchanged
      Move left on both tapes

$1^{n\%m}$ is now on Tape 1. Tape 2 is blank.
This is what is supposed to happen when a multi-tape machine computes a function.

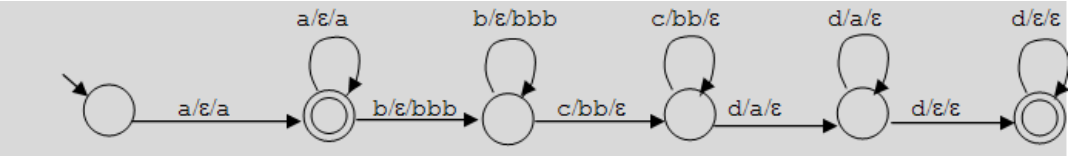
5. (15 points) Let $L = \{a^m b^{2n} c^{3n} d^p : p > m, \text{ and } m, n \geq 1\}$
   a) (3) What is the shortest string in L? abbcccdd
   b) (6) Write a context-free grammar that generates L.

Here are Two solutions (there are others):

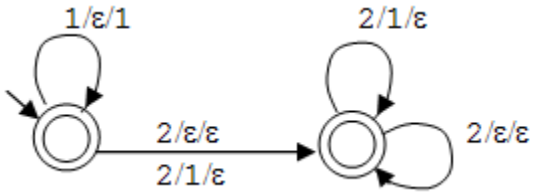
$S \rightarrow aXdd$, $X \rightarrow Xd$, $X \rightarrow aXd$, $X \rightarrow bbYccc$, $Y \rightarrow bbYccc$, $Y \rightarrow \varepsilon$, or

$S \rightarrow aSd$, $S \rightarrow Sd$, $S \rightarrow aMdd$, $M \rightarrow bbccc$, $M \rightarrow bbMccc$)

c) (6) Define a pushdown automaton that accepts L. Show a transition diagram.



6. (10 points) Consider the following PDA:



a) (3) Give a concise description of $L(M)$.

$\{1^n 2^m : 0 \le n \le m\}$ [Note: Only accepts when stack is empty]

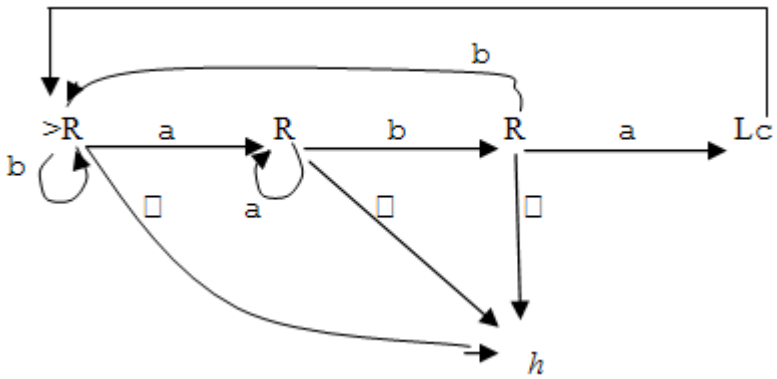
b) (3) Is M deterministic? Justify your answer.

No. Whenever there is a 1 on the stack and the input symbol is 2, the two transitions from the start state to the other state compete with each other.

c) (4) Is $L(M)$ deterministic context-free? Justify your answer.

Yes. There exists a deterministic PDA that accepts $L(M)$. It works similarly to the way M works except that, before it begins reading input, it pushes a marker # onto the bottom of the stack. Then it only takes the 2 transitions that don't pop a 1 if the stack contains no 1's. **If students givet the wrong answer for a, and give an answer here that is consistent with that answer, 2 points here.**

7. (10 points) Give a short but careful English description of what this TM does.



It replaces each occurrence of *aba* in its input string by *aca*,
4 points if student gives answer "replaces all a's with c's".

8. (5 points) Where does the "k" in the Pumping Theorem for context-free languages come from?
[Hint: for a regular language, k is the number of states in a DFSM that recognizes the language.]

If the language is CF< it has a CFG. Let b be the "branching factor", the maximum length of the RHS of any production. Let N be the number of different nonterminals. Then k <= N^{b+1}