MA/CSSE 474 Exam 3 Winter 2013-14 Name___**Solution**_with explanations_____ Section: 02(3rd) 03(4th)

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, 0 for leaving it blank, and -1for circling the incorrect answer. In order to 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 Some languages are not decidable, but every language is semi-decidable.
   H is in SD but not in D. If ¬H were in SD also, then we could simultaneously run TMs for H and ¬H. If H machine halts and accepts, accept; if ¬H machine halts and accepts, reject. One or the other must halt and accept. Thus H would be decidable. But it's not!

   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 every context-free language is non-context-free. Σ* and ∅ are both CF.

   d) **T** F IDK Every context-free language is decidable. Theorem 14.1

   e) **T** F IDK The set of decidable languages is closed under complement.
   If L is decidable, there is a TM M that always halts and ends up in stare y or n. Make a TM M' that is identical to M except reverse the positions of y and n. M' decides ¬L.

   f) T **F** IDK The set of semidecidable languages is closed under complement. See part a.

   g) T **F** IDK The set of non-semidecidable languages is closed under complement. Consider ¬H, as in part a.

2. (30 points) For each of the following languages, circle
   R if the language is Regular,
   CF if it is Context-free but not Regular,
   D if it is Decidable but not Context-free
   SD if it is Semidecidable but not Decidable
   ¬SD if it is *not* Semidecidable
   IDK if you don't know. **Scoring:** Correct answer 3, IDK 1, incorrect answer 0.

   a) R **CF** D SD ¬SD IDK $WW^R = \{ww^R : w \in \{a,b\}^*\}$. Example 11.3

   b) **R** CF D SD ¬SD IDK { <M> : where M is a TM with 5 states, and tape alphabet {☐, a}}. It's finite.

   c) **R** CF D SD ¬SD IDK $\{a^n : n \geq 0\}$. a* is a reg. exp. For it.

   d) R **CF** D SD ¬SD IDK $\{a^n b^n : n \geq 0\}$. Examples 8.8, 11.2

   e) R **CF** D SD ¬SD IDK $\{a^n b^n b^n : n \geq 0\}$. $S \rightarrow \varepsilon \mid aSbb$. Easy to use Pumping Theorem to show non-regular.

   f) R CF D **SD** ¬SD IDK {<M, w> : TM halts when started with input w}. Theorems 19.1, 19.2

   g) R CF D SD **¬SD** IDK {<M, w> : TM does not halt when started with input w}. See problem 1, part a

   h) **R** CF D SD ¬SD IDK $\{wxw^R : w,x \in \{a,b\}^+\}$. **Reg exp.: a(a∪b)a ∪ b(a∪b)b**

   i) **R** CF D SD ¬SD IDK $L(G)$, where $G = \{S \rightarrow TSb \mid S \rightarrow Tb, T \rightarrow Ta, T \rightarrow \varepsilon\}$. It's {a}*

   j) R **CF** D SD ¬SD IDK $\{x\#y\#z : x, y, z \in \{a, b\}^+ \text{ and } (|x| - |y| = 2 \text{ or } |y| - |z| = 2\}$.

   $S \rightarrow A \# X \mid X \# B$
   $A \rightarrow C A C$ /* $|x| - |y| = 2$
   $A \rightarrow C C C \# C$
   $B \rightarrow C B C$ /* $|y| - |z| = 2$
   $B \rightarrow C C C \# C$
   $C \rightarrow a \mid b$
   $X \rightarrow C X \mid X C \mid C$

L is not regular, which we show by pumping. Let $w = a^{k+2}\#a^k\#a^{k+2}$. y must occur in the first a region and must be a^p for some nonzero p. Pump in. The three regions of the resulting string are now of the following lengths:

x: k + 2 + p
y: k
z: k + 2

So neither of the required conditions is met and the resulting string is not in L.

3. (20 points) Let *G* be the grammar S → SS | (S) | ε.
L(*G*) is the language *Bal* of all strings of balanced parentheses; that is, those strings that could appear in a well-formed arithmetic expression. We may want to show that L(*G*) = *Bal*, which requires two inductive proofs:

If *w* is in L(*G*), then *w* is in *Bal*.
If *w* is in *Bal*, then *w* is in L(*G*).

We shall here prove only the first part. You will see below the sequence of steps in the proof, each with the reason for its validity omitted. These reasons belong to one of three classes. You do not have to write out reasons; just **correctly choose and write A, B, or C in each blank**. There are ten such blanks.

A) Use of the inductive hypothesis.

B) Reasoning about properties of grammars, e.g., that every derivation has at least one step.

C) Reasoning about properties of strings, e.g., that every string is longer than any of its proper substrings.

The proof is by induction on the number of steps in the derivation of *w*.

Basis: One step. The only 1-step derivation of a terminal string is S => ε because (B) this production is the only one with only terminals in the body.

ε is in *Bal* because (C) the empty string trivially has only balanced parenthesis. It is, for example, the "parentheses" in an expression like x that has no parentheses.

Induction: An n-step derivation for some n>1.

The derivation S $=>^{n}$ *w* is either of the form

(a) S => SS $=>^{n-1}$ *w* or of the form

(b) S => (S) $=>^{n-1}$ *w*

because _ (B) these are the only productions that allow further steps, i.e., they have at least one nonterminal in the body.

Case (a):

w = *xy*, for some strings *x* and *y* such that S $=>^{p}$ *x* and S $=>^{q}$ *y*,

where p<n and q<n because (B) two adjacent nonterminals in a sentential form generate adjacent terminal strings and share the steps between them in some order.

x is in *Bal* because (A) the inductive hypothesis.

y is in *Bal* because (A) the inductive hypothesis

w is in *Bal* because (C) the concatenation of strings with balanced parentheses also has balanced parentheses.

Case (b):

w = (*z*) for some string *z* such that S $=>^{n-1}$ z because (B) a nonterminal generates a terminal string that sits between whatever comes from the symbols to the left and right of that nonerminal.

z is in *Bal* because (A) the inductive hypothesis

w is in *Bal* because (C) a pair of parentheses surrounding a string with balanced parentheses is itself balanced.

4. (10 points) What are the two ways that a nonterminal symbol in a CFG can be *useless*?

a) Unreachabe from the start symbol

b) **Can not generate any strings of only terminal symbols**

5. (10 points) Here is a context-free grammar G:

$S \rightarrow AB \mid CD$
$A \rightarrow BG \mid 0$
$B \rightarrow AD \mid \varepsilon$
$C \rightarrow CD \mid 1$
$D \rightarrow BB \mid E$
$E \rightarrow AF \mid B1$
$F \rightarrow EG \mid 0C$
$G \rightarrow AG \mid BD$

$\boxed{B\ (B \rightarrow \varepsilon),\ D\ (D \rightarrow BB),\ G\ (G \rightarrow BD),\ A\ (A \rightarrow BG),\ S (S \rightarrow AB)}$

In the list below, Circle each nullable symbol of G and do not circle non-nullable symbols.

[✓ S] [✓ A] [✓ B] C [✓ D] E F [✓ G] 0 1

6. (15 points) Here are eight simple grammars, each of which generates an infinite language of strings. These strings tend to look like alternating *a*'s and *b*'s, although there are some exceptions, and not all of the grammars generate all such strings.

a)	$S \rightarrow abS \mid ab$	(ab)*
b)	$S \rightarrow SS \mid ab$	(ab)*
c)	$S \rightarrow aB;\ B \rightarrow bS \mid a$	a(ba)*a
d)	$S \rightarrow aB;\ B \rightarrow bS \mid b$	(ab)*
e)	$S \rightarrow aB;\ B \rightarrow bS \mid ab$	a(ba)*ab
f)	$S \rightarrow aB \mid b;\ B \rightarrow bS$	(ab)*b
g)	$S \rightarrow aB \mid a;\ B \rightarrow bS$	(ab)*a
h)	$S \rightarrow aB \mid ab;\ B \rightarrow bS$	(ab)*

For each of the following grammar pairs, circle Y if those two grammars generate the same language, and N if they do not.

Y [✓ N] a and f

[✓ Y] N a and b

Y [✓ N] c and f

Y [✓ N] d and g (I intended to write "d and h" and have the answer be y, so if there is a bit of a mess on your paper …)

Y [✓ N] e and h

7. (15 points) The language L = {*ss* | *s* is a string of *a*'s and *b*'s} is not a context-free language. In order to prove this, we need to show that for every integer *k*, there is some string *w* in L, of length at least *k*, such that no matter how we break *w* up as $w=uvxyz$, subject to the constraints $|vxy| \leq k$ and $|vy| > 0$, there is some $q \geq 0$ such that uv^qxy^qz is not in L. Let us focus on a particular *k*=7 and *w*=*aabaaaba*. It turns out that this is not a good choice of *w* for *k*= 7, since there are some ways to break *w* up for which we can find the desired *q*, and other ways of breaking it up for which we cannot. Identify from the list below the choices of *u*,*v*,*x*,*y*,*z* for which there is an *q* that makes uv^qxy^qz **not** be in L. We show the breakup of *aabaaaba* by placing four |'s among the *a*'s and *b*'s. The resulting five pieces (some of which may be empty) are the five strings. For instance, aa|b||aaaba| means *u*=aa, *v*=b, *x*=ε, *y*=aaaba, and *z*=ε.

For each way of breaking up the string, circle Y if this way of breaking up the string allows pumping to take us outside of L, and N if every choice of q keeps the pumped string in L.

Y [✓ N] aab|a|aab|a|

Y [✓ N] aa|ba|aa|ba|

Y [✓ N] |aa|b|aa|aba

[✓ Y] N aab|a|a|a|ba pump out (q=0) to get aababa

Y [✓ N] |a|abaa|a|ba

8. (8 points) Consider the pushdown automaton with the following transition rules:

1. $\delta(q, \varepsilon, \varepsilon) = \{(q, Z_0)\}$
2. $\delta(q,0,Z_0) = \{(q,XZ_0)\}$
3. $\delta(q,0,X) = \{(q,XX)\}$
4. $\delta(q,1,X) = \{(q,X)\}$
5. $\delta(q,\varepsilon,X) = \{(p,\varepsilon)\}$
6. $\delta(p,\varepsilon,X) = \{(p,\varepsilon)\}$
7. $\delta(p,1,X) = \{(p,XX)\}$
8. $\delta(p,1,Z_0) = \{(p,\varepsilon)\}$

The start state is *q*. For which of the following inputs can the PDA enter state *p* for the first time with the input empty and the stack containing XXZ_0 [i.e., the configuration (p,ε,XXZ_0)]? Mark the correct such input. Only one is correct.

___ 0101010

_ **X** 011011011

___ 011001101

___ 1001101

When in state q, the PDA adds an X to the stack whenever it consumes a 0. The PDA may consume a 1 with no change to the stack, but only if the stack has top symbol X. That is, on inputs beginning with 1 the PDA has no choice of move and can never enter state p. Since entering state p pops an X from the stack, there must be exactly three 0's in the consumed inputs, and any number of 1's. In addition, the first input symbol must be 0.

9. (7 points) Start with the grammar G: **See top of p 262.**

$S \rightarrow AS \mid A$

$A \rightarrow 0A \mid 1B \mid 1$

$B \rightarrow 0B \mid 0$

If we use the nondeterministic top-down parsing algorithm from class (and the textbook) to convert this grammar to a PDA, which of the following transitions are part of the PDA? Circle Y if the given transition is part of the PDS, and N if it is not. p is the start sate, q is the accepting state (don't forget that in order to accept a string w, the stack must also be empty after reading w).

[✓ Y] N $((p, \varepsilon, \varepsilon),(q, S))$

Y [✓ N] $((p, \varepsilon, S), (q, \varepsilon))$

[✓ Y] N $((q, \varepsilon, S), (q, AS))$

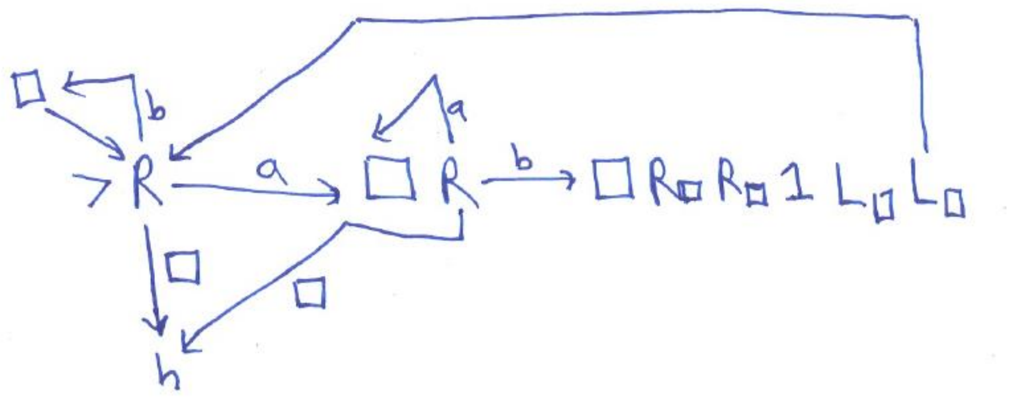
Y [✓ N] $((q, \varepsilon, A), (q, A0))$

[✓ Y] N $((q, \varepsilon, A), (q, 1))$

[✓ Y] N $((q, 1, 1), (q, \varepsilon))$

Y [✓ N] $((q, 1, \varepsilon), (q, 1))$

10. (20) Consider the following TM M. Try to figure out what M does when started with a string of a's and b's (you do not have to *describe* what it does in general, but understanding that will help you answer the questions. For each of the two inputs below, if M is started with that input on its tape, show the tape contents after M halts.



Recall that a TM always starts with its read/write head at the last blank square before the input string.
The machine "blanks out" a's and b's, but whenever it finds a substring ab, it goes all the way right and adds a 1 at then end of the string. When it halts, the tape contains a 1 for every ab substring in the original input.

Input: baabaaabaaaab Output: ______**1 1 1**___________

Each a and b gets erased when it is encountered. Every time there is an *ab* in the input, we move right and write a 1. So the machine counts (in unary) the number of times that *ab* occurs in the input string.

Input: abbabbbabbbba Output: ______**111**______________

11. (7 points) Where does the "k" in the statement of 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.]

A brief answer (two or three sentences) is probably sufficient to let me know you have the idea..

Let G, a grammar for L, have N nonterminals and branching factor b. If a string in L(G) is longer than b^N, there must be at least one path from the root of the parse tree that has a repeated nonterminal; subtrees of these nonterminals is the basis for the parts that get pumped.