More CFG Challenge Problems

1. What are the complements of L = $\{0^n 1^n \mid n \geq 0\}$?

To write a grammar for the complement of L, we first figure out what L looks like and tackle it case by case. First, we have strings that are ordered nicely (zeros followed by ones), but have an imbalanced number (e.g., a few too many zeros, or ones). Then, we have strings with the wrong order, which are strings of all kinds with at least one "swap" somewhere (a one followed by zero). Many swaps are fine also, but just one will do to make the complement.

So, L looks like:

L = A ∪ B
A = $\{0^n 1^m \mid m \neq n\}$
B = { (0+1)*10(0+1)* }

Now, we can build a grammar for L piece-by-piece and join them together.

For A, we can stuff-in extra 0's or 1's (in the right order) to "break" otherwise perfectly ordered and balanced strings:

A → 0 A 1 | 1 X | 0 Y
X → 1 X | ϵ
Y → 0 Y | ϵ

For B, we simply need a swap ("10") surrounded by the any string generator:

B → Z 1 0 Z
Z → 0 Z | 1 Z | ϵ

Finally, join together for the grammar giving the language L:

S → A | B
(...etc.)

2. L = {w#x | w^R is a substring of x} is context free, but M = {w#x | w is a substring of x} is not context free!

For L, strings look like, for example (with w and w^R bolded):
0001#1010101**1000**101011

Thus, L is of the form: w#(0+1)*w^R (0+1)*. We already have our any-string generator, X, from above:

X → 0 X | 1 X | ϵ

What we need now is the palindrome piece, which we have seen already:

P → 1 P 1 | 0 P 0

We modify it slightly to stuff-in the "#(1+0)*" portion once the palindrome is done generating:

P → 1 P 1 | 0 P 0 | # X

Finally, we tac these together to give us the full string:

S → P X
(.... etc.)