1. (30 points) Let $G$ be the CFG below:

$S \rightarrow AABB$

$S \rightarrow ABAB$

$A \rightarrow AAA$

$B \rightarrow BBB$

$A \rightarrow a$

$B \rightarrow b.$

Recall that $L(G)$ is the set of all strings that are generated by $G$.

(a) (15 points)

Give an infinite number of strings that are NOT in $L(G)$.

(b) (15 points) RECALL that Chomsky Normal Form Grammars have rules of the form

$X \to YZ$ (for any NTs $X, Y, Z$)

$A \to \sigma$ (for any NT $A$ and any $\sigma \in \{a, b\}$.

$S \to e$ (this one you won't need).

For this problem we also allow rules of the following form to make your life easier

$X \to Y$ (for any NTs $X, Y$).

Give a Chomsky Normal Form Grammar $G'$ such that $L(G) = L(G')$. You should use the algorithm I discussed in class in the Feb 25,27 CFG lecture.

2. (40 points) You can and should look on the web for the answer to Part $a$ of this question.

Put it in you own words.

Give a CLEAR explanation- since we are not testing your knowledge or cleverness we are testing how well you can express yourself.

Make sure you understand the construction and COULD write a program to do it OR do it by hand.

(a) (20 points) Give an algorithm that will,
GIVEN a DFA $M = (Q, \Sigma, \delta, s, F)$ produce
a CFG $G = (V, \Sigma, R, S)$ such that $L(M) = L(G)$.
For the CFG you must specify

- $V$ The set of nonterminals.
- $S$ The Start nonterminal.
- $R$ The rules.

We want a DIRECT construction- DO NOT go from DFA to regex to CFG.

(b) (0 points but you need this for the next problem.) Write a DFA for
$L = \{w \colon \#_a(w) \equiv 0 \pmod 2 \wedge \#_b(w) \equiv 0 \pmod 2\}$.
(You can either give a diagram or a table.)

(c) (20 points) Give a CFG for $L$ by using the algorithm in part a on the DFA in part b.

(d) (0 points) THINK ABOUT but do not hand in:
Let $L_n = \{w \colon \#_a(w) \equiv 0 \pmod n \wedge \#_b(w) \equiv 0 \pmod n\}$.
In a prior HW I asked how big a CFG for this problem would be. It was quite large since you went
DFA to regex to CFG.
NOW that we an go DFA to CFG, how large will the CFG for $L_n$ be?

3. (30 points) In this problem we allow CNF grammars to have rules of the following types

$A \to BCD$

$A \to BC$

$A \to \sigma$

$A \to e$

**PREAMBLE**

(a) (0 points but it gives the problem context) Fill in the blank.
In class we showed that, for all $t$, the langage

$$\{e, a, a^2, \ldots, a^{t^3-1}\}$$

has a CNFG with FILL IN nonterminals (you CANNOT use O-notation).

(b) (0 points but it gives the problem context) Fill in the blank.
In class we showed that, for all $n$, the langage

$$\{e, a, a^2, \ldots, a^n\}$$

has a CNFG with FILL IN nonterminals (you CAN use $O$-notation).

**END OF PREAMBLE**

The acutal problem is on the next page.

(a) Here is a CFG

$E_0 \to a \mid e$ (so $L(E_0) = \{e, a\}$.

$E_1 \to E_0 E_0 \mid e$ (so $\{e, a^2\} \subseteq L(E_1)$).

$E_2 \to E_1 E_1 \mid e$ (so $\{e, a^4\} \subseteq L(E_2)$).

$E_3 \to E_2 E_2 \mid e$ (so $\{e, a^8\} \subseteq L(E_3)$).

$E_4 \to E_3 E_3 \mid e$ (so $\{e, a^{16}\} \subseteq L(E_4)$).

$E_5 \to E_4 E_4 \mid e$ (so $\{e, a^{32}\} \subseteq L(E_5)$).

$E_6 \to E_5 E_5 \mid e$ (so $\{e, a^{64}\} \subseteq L(E_6)$).

$S \to E_0 E_1 E_2 E_3 E_4 E_5 E_6$

i. (10 points) Show that $L(G) = \{e, a, a^2, \ldots, a^{127}\}$ (Hint: Use that every number can be written in base 2.)

ii. (10 points) Convert the grammar to CNF. Just have to change the $S$ rule into a set of rules that do the same thing but are in CNF. How many nonterminals does it have? (We allow rules of the form $A \to e$ so this is not strictly CNF.)

iii. (10 points) In the following statemeng FILL IN and prove it. Use the grammar above as a guide.
*For all $t$ there exists a CNFG $G$ that generates $\{e, a, a^2, \ldots, a^{2^t - 1}\}$ that has FILL IN number of nonterminals. You may use O-notation.*

iv. (0 points but its easy and finishes up this line of reasoning.) In the following statemeng FILL IN and prove it. Use the grammar above as a guide.
*For all $n$ there exists a CNFG $G$ that generates $\{e, a, a^2, \ldots, a^n\}$ that has FILL IN number of nonterminals. You may use O-notation.*