1. For each of the following statements indicate whether it is *true* or *false*. For the *false* ones (if any), provide a counter example. For the *true* ones (if any) give a proof outline.

   (a) Union of two non-regular languages cannot be regular.
   Ans: False.
   Let $L_1 = \{a^m b^n \mid m \geq n\}$ and $L_2 = \{a^m b^n \mid m < n\}$ and $L_1 \cup L_2 = a^*\ b^*$ which is regular.

   (b) Union of a regular language with a disjoint non-regular language cannot be regular.
   Ans: True.
   Let L_1 is a *regular* language and L_2 is a *non−regular* language and they are disjoint i.e. $L_1 \cap L_2 = \emptyset$. Suppose $L = L_1 \cup L_2$ and L is *regular* (since *regular* languages are closed under union).
   Now $\overline{L_1}$ is *regular* (since *regular* languages are closed under complementation). Since, $\overline{L_1}$ is *regular*, hence its intersection with L i.e $\overline{L_1} \cap L = L_2$ is *regular* (since *regular* languages are closed under intersection). Therefore, L_2 is *regular*. Hence Contradiction.

   (c) The set of finite length strings over a countably infinite alphabet is countably infinite.
   Ans: True.
   Consider S_l as the set of strings of a fixed finite length l. Each string in this set can be considered as an element of Σ^l, which is a finite product of countably infinite sets, and therefore, countably infinite. S_l is therefore, countable. Consider the set $\bigcup_{l \geq 0} S_l$. Now you have a finite union of countably infinite sets, which is again countably infinite.

   (d) $L((ab^*ba^*) \cap (ba^*ab^*)) = \epsilon$
   Ans: False.
   It is NULL.

   (e) The symmetric difference SD of 2 languages (over some alphabet Σ) is always context-free. SD(L,M) of languages L and M is the set of strings in exactly one of L and M.
   Ans: False.
   Counterexample: Let $M_1 = L(0^*1^*2^*)$ and $M_2 = \{0^i 1^j 2^k \mid i \neq j \text{ or } i \neq k\}$. $SD(M_1, M_2)$ is the known non-CFL $\{0^n 1^n 2^n \mid n \geq 0\}$.

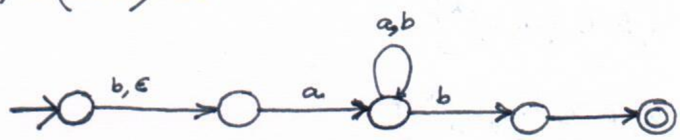
   (f) The intersection of two context free languages is never context free.
   Ans: False.
   Counterexample: Regular languages.

   (g) Every finite subset of Σ^* is a regular set..
   Ans: True.
   Let the finite language be $L = \{x_1, x_2, \ldots x_k\}$. This can be generated by the right-linear grammar S $\rightarrow x_1 \mid x_2 \mid \ldots \mid x_k$.

   (h) Let L = $\{1^n\ : n \leq 1000$ and n is prime. A DFA accepting L may have less than 900 states.
   Ans: False.
   The prime nearest to 1000 is 997. To detect this, the machine cannot have less than 900 states, since no loop is possible other than the sink state.

2(~~b~~a) $(b \cup \epsilon)\, a\, (a \cup b)^*\, bb$

b)



(ii) Let $M = (Q, \Sigma, \delta, s_0, F)$ be a DFA accepting L. We design an ϵ-NFA $M' = (Q', \Sigma, \delta', s_0', F')$ to accept odd(L). Let $Q = \{p_1, p_2, \dots, p_k\}$. Then we take $Q' = \{p_1, p_2, \dots, p_k, q_1, q_2, \dots, q_k\}$, that is, we add a new state q_i for each p_i. The only start state of M′ is the old start state of M, i.e. $S_0' = S_0$. All the ~~new~~ final states of M continue to remain final in M′. Moreover, a new state q_i is marked final in M′ if and only if the corresponding old state p_i is final (in M). We replace each transition $p_i \xrightarrow{a} p_j$ in M (where $a \in \Sigma$ and where we may have, $i = j$) by the two transitions

$$p_i \xrightarrow{a} q_j \quad \text{and} \quad q_i \xrightarrow{\epsilon} p_j$$

By construction, moves in M′ alternate between the old and the new states. In essence, M′ mimics M with the only exception that every second move prevents consuming a real symbol from the input.

3(i) Let l be the length of the longest string in L. Since L is given to be finite, l is finite, too. We also have $m \le 1 + 2 + 2^2 + \dots + 2^l = 2^{l+1} - 1$ i.e. $l + 1 \ge \log_2 (m+1)$. If $\alpha \in L$ ~~and~~ is of length l and if $l \ge n$, then by the pumping lemma, we have strings $\beta_1, \beta_2, \beta_3 \in \Sigma^*$ such that $|\beta_2| \ge 1$ and $\beta_1 \beta_2^k \beta_3 \in L$ for all $k \in Z_+$, implying L is infinite, a contradiction. Therefore, $l < n$ i.e. $n \ge l+1 \ge \log_2 (m+1)$

(ii) (a) Yes.

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaSa \Rightarrow aaSaa \Rightarrow aaSSaa \Rightarrow aabSaa \Rightarrow aabbaa$$

(b) bab : Consider the two different leftmost derivations for this string.

$$S \Rightarrow SS \Rightarrow bS \Rightarrow baS \Rightarrow bab$$

$$S \Rightarrow SS \Rightarrow SaS \Rightarrow baS \Rightarrow bab$$

(c) False. Counterexample: $bab \in L(G)$ [Derivation above]

but $bab \notin L(a^* b b^* a^*)$

4 (a) Assume L is context-free and let n be the constant for the language L prescribed by the stronger version of the pumping lemma. Consider $\alpha := a^n c a^n c a^n \in L$. The pumping lemma gives us the decomposition $\alpha = \beta_1 \beta_2 \beta_3 \beta_4 \beta_5$ with $|\beta_2 \beta_4| \ge 1$ and $|\beta_2 \beta_3 \beta_4| \le n$. Since $\alpha' = \beta_1 \beta_3 \beta_5 \in L$, $\beta_2 \beta_4$ must not contain the symbol 'c', i.e. $\beta_2 \beta_4$ consists only of 'a's. The condition $|\beta_2 \beta_3 \beta_4| \le n$ implies $\beta_2 \beta_4$ cannot stretch over all the three runs of a's in α. Therefore, α' lacks the defining property of the strings of L. This contradiction shows L is not context-free.

4 (b) Define

$$L_1 = \{ \alpha c \alpha^R c \beta \mid \alpha, \beta \in \{a,b\}^* \}$$

$$L_2 = \{ \beta c \alpha^R c \alpha \mid \alpha, \beta \in \{a,b\}^* \}$$

Clearly, $L = L_1 \cap L_2$.

To show L_1 is context free, consider the following CFG $:= (\{S, U, V\}, \Sigma, S, R)$ for L_1 where the productions are:

$$S \to UV$$

$$U \to c \mid aUa \mid bUb$$

$$V \to c \mid Va \mid Vb$$

An analogous CFG defines L_2.

5. A *wiggle string* is defined to be a nonempty string of 0's and 1's such that each 0 is followed by a 1, and each 1 is followed by a 0. A *0-wiggle string* is a wiggle string that begins and ends in 0, and a *1-wiggle string* is a wiggle string that begins and ends with 1. For example, 01010 is a 0-wiggle string, 1 is a 1-wiggle string, 0101 is a wiggle string that is neither a 0-wiggle string nor a 1-wiggle string, and 010010 is not a wiggle string. The language of the grammar G consisting of productions

S -> SAS | 0
A -> ASA | 1

(S is the start symbol) is the set of 0-wiggle strings, as you will prove. In this proof, you can assume the true fact that the concatenation of two wiggle strings is a wiggle string, as long as the first ends in a symbol different from the symbol by which the second begins. You do not have to state this fact in the proof. Give your proof by answering the following very specific questions.

Prove first that every 0-wiggle string is in L(G). To do so, we need to prove a more general statement inductively: (1) if w is a 0-wiggle string then S =>* w, and (2) if w is a 1-wiggle string, then A =>* w.

(a) On what is your induction?

The length of w.

(b) What is the basis case?

|w| = 1.

Comment: *The induction is always on some parameter, and the basis case is always an integer or set of integers.*

(c) Prove the basis.

(1) w = 0, and S => 0. (2) w = 1, and A => 1.

(d) For the inductive part, first show that if w is a 0-wiggle string, then S =>* w.

Assume |w| = n > 1 and assume the IH for strings shorter than length n. If w is a 0-wiggle string, then w = 01x for some 0-wiggle string x.
By the IH, S => x. Thus, S => SAS => 0AS => 01S =>* 01x = w.*

The second part of the induction is that if w is a 1-wiggle string, then A =>* w. You do not have to provide this part, since its proof is essentially like that of (d), with 0 and 1 interchanged.

Conversely, to show that every string in L(G) is a 0-wiggle string, we shall show that (1) if S =>* w, then w is a 0-wiggle string, and (2) if A =>* w, then w is a 1-wiggle string.

(e) On what is your induction?

Number of steps in the derivation.

(f) What is the basis case?

One step.

(g) Prove the basis.

The only one-step derivations are S => 0 and A => 1. The resulting strings are 0- and 1-wiggle strings, respectively.

(h) For the inductive part, first show that if S =>* w, then w is a 0-wiggle string of length n.

Assume the derivation is multistep, and assume the IH for shorter derivations. Then the derivation begins S => SAS => w. We can break w = xyz, such that in this derivation S =>* x, A =>* y, and S =>* z, all by shorter derivations. By the IH, x and z are 0-wiggle strings, and y is a 1-wiggle string. Thus, by the properties of wiggle strings stated in the opening of the problem, xyz = w is a 0-wiggle string.*

To complete the inductive part, you also need to show that if A =>* w, then w is a 1-wiggle string. You do not need to provide this part, since it is essentially (h), with 0 and 1 interchanged.