A solution to Assignment 3

1. The language L_a is regular but L_b and L_c are not.
   (a). For thi swe provide a FA for L_a. Let M be a DFA for the regular language L used to define L_a. We then duplicate M and construct an FA M' for L_a as follows. Let us call these two copies of M as M1 and M2. Place M1 below M2. Connect every state q_i in M1 to its corresponding state q_i in M2 with a new transition with label a. Remove the initila state designation from q_0 in M2 and let the initial state q_0 in M1 be the initial state of M'. Remove the final state designaton from every final state in M1. The final state(s) in M2 will be the final state(s) of M'. If w is any string in L with w=uv for all substrings u and v in Σ^*, then M' processes substring u through M1 (the upper copy of the DFA M) and reaches some state q_j. The process then continues following the transition with label a added from q_j to its corresponding state q_j in M2, (the lower copy of M) and then substring v will be processed through the lower part and ends in some final state. That is, if the DA M accepts w, the FA M' defined above accepts the string uav, for all substring u and v such that w=uv.

   (b). We will show that the set L_b of palindromes is not regular. Suppose it is regular. Then the exists a DFA M that accepts it. Furthermore, since L_b is infinite, then the pumping lemma applies. Let m be the number of states in M. Let us consider the string $w = a^m b^m b^m a^m$ in L_b, whose length $(4m)$ is $\geq m$. Then, by P.L., w can be decomposed into substrings x, y, z such that (1) $|xy| \leq m$ (2) $|y| \geq 1$ such that for all $i \geq 0$, string $w_i = xy^iz$ is in L_b. This implies that the substring y consists of a's only and in the first m suymbols. That is $y = a^k$, for some $1 \leq k \leq m$. Now we pick $i = 0$, the string $w_0 = a^{m-k} b^m b^m a^m$ is not in L_b but is accepted by M, which is a contradition, ane hence there is no DFA exists for L_b, which means L_b is not a regular language.

   (c). Suppose L_c is a regular language. Then the exists a DFA M such that $L = L(M)$. Since L_c is infinite, the requirement specified in the pumping lemma applies. Let m be the number of states in M. Consider the string $w = a^m b^m$ in L_c, whose length $2m$ is $\geq m$. Then, by P.L., there ar esubstrings x, y, z such that $w = xyz$ such that $|xy| \leq m$ and $|y| \geq 1$ such that for every $i \geq 0$, the string $w_i = xy^iz$ is in L_c. This implies that y consists of a's only, i.e., $y = a^k$, for some $1 \leq k \leq m$. If we pick $i = 2$, the string $w_2 = a^{m+k}b$ will also be accepted by M however it does not belong to L_c. This means M is not a DFA for L_c, and since we did not make specific assumption about M, this implies no DFA exists for L_c, and hence it is not regular.
2. Let L be any CFL. Then there exists a CFG G that generates L. We modify G to obtain the CFG G' as follows. For every production $A \rightarrow v$ in G, we

include in G' the production $A \to v^R$, where v^R is the reverse of the string v. We can use the induction technique on the length of strings w in L to show that if $w \in L(G)$, then $w^R \in L(G')$. This establishes that CFLs are closed under reversal.

Note: The argument could be made done easier if the CFG G we consider for L is in Chomskey Normal Form (CNF). However, if λ is in L, we consider a CFG grammar G'' in CNF for $L - \{\lambda\}$ and then add the production $S \to \lambda$ to the modified grammar G' obtained to be equivalent to G''.

3. (a). (i) Removing λ-productions. Since λ is generated by G, we can give a CFG in CNF only for $L(G) - \{lambda\}$. Substituting $A \to \lambda$ we get two new productions, $A \to aa$ and $S \to \lambda$. Substituting the production $S \to \lambda$ has no further effect, however as explained, we do not consider it further.

   (ii) Removing unit-productions. We draw the dependency graph for the variable S,A,B, and C involved in unit productions. Analyzing the dependency graph, we find that $S \Rightarrow^* A$, $S \Rightarrow^* B$, and $C \Rightarrow^* B$. Using Using this information, we remove $S \to A$ and add $S \to aaA$, we remove $S \to B$ and add $S \to bB \mid bbC$, and finally replace $C \to B$ with $C \to bB \mid bbC$.

   This yields the following grammar:

   $S \to aa \mid aaA \mid bB \mid bbC$
   $A \to aa \mid aaA$
   $B \to bB \mid bbC$
   $C \to bB \mid bbC$

   (iii) Removing useless productions. Note that the only useful variables are S and A; B and C are useless and hence every production that involves B or C is removed. This completes the preprocessing phase which yields the following CFG:

   $S \to aa \mid aaA$
   $A \to aa \mid aaA$

   We are now ready to convert the resulting grammar above into CNF:

   $S \to XX \mid XXA$
   $A \to XX \mid XXA$
   $X \to a$

   We introduce new variables X and Y to break the right hand side of productions which have more than two variables. This yeilds the following grammar in CNF for $L(G) - \{\lambda\}$.

   $S \to XX \mid XY$
   $Y \to XA$
   $A \to XX \mid XY$
   $X \to a$

   Remark: Note that the above grammar generates the language $\{a^{2n} : n \geq 1\}$, which is regular. A regular grammar for this lannguage is $S \to aaS \mid aa$.

4. Assume that $\lambda \notin L(G)$. We will discuss later if this is not the case. Since G is a CFG, based on Theorem 6.7 on page 176 in the textbook, there is an equivalent CFG G' in Greibach normal form. That is, each production in G' is of the form $A \to \alpha v$ where $\alpha \in \Sigma$ and $v \in V^*$, i.e., v consists of

variables only. We will show how to transform such productions into the required forms in this question.
Possible productions in G' (which is in Greibach NF) would be:
$(1a)$ $A \to a$ $(|v| = 0)$
$(1b)$ $A \to aB$ $(|v| = 1)$
$(1c)$ $A \to aBC$ $(|v| = 2)$
$(1d)$ $A \to aBCD$ $(|v| = 3)$
$\cdots$
For each of these production types, we describe below how to transform it into the required form.
For every production of type $1a$, we introduce a new variable, say V_i, and replace production of this type with the following two rules:

$A \to aV_iV_i$
$V_i \to \lambda$

For each production of type $1b$, we introduce a new variable V_2 and replace $1b$ with the following two productions:

$A \to aBV_j$
$V_j \to \lambda$

For productions in type $1c$, they are already in a desired form.
For type $1d$, we introduce a new variable V_k and replace the rule with:

$A \to aBV_k$
$V_k \to CD$

In general, for productions of the form $A \to aB_1B_2 \cdots B_n$, we introduce new variables $C_1, \cdots, C_{n-1}$ and replace these productions with the following ones:

$A \to aB_1C_1$
$C_1 \to B_2C_2$
$C_2 \to B_3C_3$
$\cdots$
$C_{n-1} \to B_{n-1}B_n$

We next consider the case when $\lambda \in L(G)$. For this we first consider the language $L(G) - \{\lambda\}$ and perform the above transformation. Recall that this is possible since $L(G) - \{\lambda\}$ does not include λ so we can get an equivalent CFG G' in Greibach NF.
Nest, we introduce a new start variable S' and add the rule $S' \to S$ to the resulting grammar in which S is the start variable. This newly added production is not in the required form. To fix this, we introduce a new variable X and replace $S' \to S$ with the following three rules:

$S' \to SX \mid \lambda$
$X \to \lambda$

5. The languages given in parts (1), (2) and (5) are CF, but (3) and (4) are not. Below we give CFGs for 1,2,and 5.

(1).
$S \to AB$
$B \to aBa \mid bBb \mid aAa \mid bAb$
$A \to aa \mid ab \mid ba \mid bb$

(2). $S \to aSa|bSb|a|b|\lambda$

(5). /*variable O is for odd length strings and E for even length */

$$S \to AcB \mid BcA \mid OcE \mid EcO$$
$$A \to aAa \mid aAb \mid bAa \mid bAb \mid a$$
$$B \to aBa \mid aBb \mid bBa \mid bBb \mid b$$
$$O \to aOa \mid aOb \mid bOa \mid bOb \mid a \mid b$$
$$E \to aEa \mid aEb \mid bEa \mid bEb \mid \lambda$$