Problem 1

Let $L$ be the set of all strings

$$a^{n_1}b\cdots a^{n_t}b \in \Sigma^* = \{a, b\}^*$$

where $t, n_1, \ldots, n_t \geq 0$ are positive integers such that $n_j \neq j$ for some $j \in \{1, \ldots, t\}$. (As usual, $a^{n_i}$ denotes the strings consisting of $n_i$ copies of the letter “a”. ) In this problem you will prove that $L$ is a context free language. You can break your proof into the following subproblems:

- First prove that the language $L' = \{c^n d^m e : n + 1 \neq m\}$ over the alphabet \{c, d, e\} is context free giving a context free grammar that generates $L'$.

**Solution:** The language $L'$ is generated by the following grammar:

\[
S \rightarrow Te|Ue \\
T \rightarrow cTd|cT|\epsilon \\
U \rightarrow c Ud| Ud| dd
\]

where $T$ generates the set of all strings $c^n d^m$ with $m \leq n$ and $U$ generates the set of all strings $c^n d^m$ with $m \geq n + 2$.

- Then consider $c, d, e$ as variable symbols and extend the grammar from the first part with rules for $c, d, e$ in such a way that the new grammar generates the language $L$.

**Solution:** We add rules to the grammar in such a way that symbols $c, d, e$ are replaced by the languages corresponding to regular expressions $a^*b$, $a$, $b(a^*b)^*$. For clarity, we rename symbols $c, d, e$ with the corresponding uppercase letters $C, D, E$. The resulting grammar is:

\[
S \rightarrow TE|UE \\
T \rightarrow CTD|CT|\epsilon \\
U \rightarrow CUD|UD|DD \\
C \rightarrow aC|b \\
D \rightarrow a \\
E \rightarrow b|EX \\
X \rightarrow aX|b
\]
The language generated by the grammar is the set of all strings obtained by concatenating $n$ strings of the form $a^*b$ (corresponding to the $C$ variable symbols), a string $a^m b$ where $m$ is an integer different from $n+1$ (corresponding to the $D$ symbols and the first $b$ in the string generated by $E$), and finally any number of strings of the form $a^*b$.

Problem 2
Transform the following PDA into an equivalent grammar.

![Diagram](image)

Figure 1: Push down automaton $M$

**Solutions:** First of all notice that whenever the automaton shown in the picture reaches the accepting state, the stack is always empty. This is because the automaton pushes only one $\$$ on the stack at the beginning of the computation, and this $\$$ is removed when the automaton goes from state 2 to state 3. Moreover, all transitions in the state diagram correspond to single push or pop operations. Therefore, we can apply the transformation procedure studied in class directly to the automaton, without adding auxiliary states to empty the stack at the end of the computation or remove simultaneous push/pop operations. We define a grammar with 9 variables $X_{ij}$ (i,j = 1,2,3), start variable $X_{13}$, alphabet \{a,b\} and the following rules. For all $i,j,k \in \{1,2,3\}$ we have rules:

\[
X_{ij} \rightarrow X_{ik}X_{kj}
\]
\[
X_{ii} \rightarrow \epsilon
\]

Moreover, since $(2,\$$) \in \delta(1,\epsilon,\epsilon)$ and $(3,\epsilon) \in \delta(2,\epsilon,\$$) we add the rule:

\[
X_{13} \rightarrow X_{22}
\]

Similarly, since $(2,a) \in \delta(2,a,\epsilon)$ and $(2,\epsilon) \in \delta(2,b,a)$ we add the rule:

\[
X_{22} \rightarrow aX_{22}b.
\]

The final grammar has $n^3 + n + 2$ rules. If we remove unnecessary rules (i.e., rules containing variables that generate the empty language), we get the simpler grammar:

\[
X_{13} \rightarrow X_{22}
\]
\[
X_{22} \rightarrow aX_{22}b|X_{22}X_{22}|\epsilon
\]
This grammar can be further simplified into
\[ S \to SS|aSb|\epsilon. \]

Notice that if we interpret \( a \) as an open parenthesis and \( b \) as a closed parenthesis, the language generated by this grammar is the set of all well-parenthesized expressions.

**Problem 3**

Let the *reverse* of a language \( L^R \) be as defined in problem 1.24 of the textbook. In this problem you will show that the reverse of a context free language is context free. Answer the following questions:

- Show how to transform any context free grammar \( G \) into another grammar \( G' \) such that the language generated by \( G' \) is the reverse of the language generated by \( G \).

**Solution:** Let \( G = (V, \Sigma, R, S) \) be a context free grammar. We define a new grammar \( G' = (V, \Sigma, R', S) \) with the same variable symbols, alphabet and start symbol. The rules \( R' \) for the new grammar are defined as follows. For each rule \( X \to Y_1Y_2\ldots Y_n \) \( \in R \) we define a corresponding rule \( X \to Y_nY_{n-1}\ldots Y_1 \) \( \in R' \). One can easily show (by induction on the length of the derivations) that a string \( Y_1\ldots Y_n \) (over \( V \cup \Sigma \)) can be generated from \( S \) in \( G \) if and only if the reversed string \( Y_n\ldots Y_1 \) can be generated from \( S \) in \( G' \). In particular the language generated by \( G' \) is the reverse of the language generated by \( G \).

- Show how to transform any PDA \( M \) into another PDA \( M' \) such that the language generated by \( M' \) is the reverse of the language generated by \( M \).

**Solution:** The simplest way to solve this problem is to use the result from the first part, and the equivalence between context free grammars and push down automata. Given a PDA \( M \), one can use the construction from Lemma 2.15 in the textbook to transform it into an equivalent grammar \( G \). Then using the first part of this problem we transform \( G \) into a grammar recognizing the reverse of \( L(G) \). Finally, we use the construction from Lemma 2.13 in the textbook to transform \( G' \) into an equivalent push down automaton \( M' \). Obviously the automaton \( M' \) recognizes the reverse of the language recognized by \( M \).

It is also possible to solve this problem directly, inverting the direction of all transitions in the push down automaton, and swapping all push and pop operations. The start state becomes the only final state. Finally, create a new start state with \( \epsilon, \epsilon \to \epsilon \) transitions to all old final states.

**Problem 4**

For each of the following languages says whether the language is context free or not, and prove it.
• $L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$ **Solution:** The language is context free because it is generated by the following grammar:

$$S \rightarrow aSc | T$$

$$T \rightarrow bTc | \epsilon$$

• $L_2 = \{a^n b^{n+1} c^{n+2} | n \geq 0\}$ **Solution:** We prove that the language is not context free using the pumping lemma for context free languages. Assume for contradiction that $L_2$ is context free and let $p$ be the pumping length. The string $w = a^p b^{p+1} c^{p+2}$ is in $L_2$ and has length bigger than $p$. Therefore, by the pumping lemma there exist substrings $wx_iz = w$ such that $v$ and $y$ are not both empty and $w^i xy^iz \in L_2$ for all $i$. Notice that $v$ and $y$ must be of the form $s^k$ where $s$ is one of $a,b,c$ because otherwise the string $w^2 xy^2 z$ would not be of type $a^* b^* c^*$. So, assume without loss of generality that $v = a^n$ and $y = b^m$ (the other cases are similar). Then the string $w^2 xy^2 z = a^{p+n} b^{p+m+1} c^{p+2}$ belongs to $L_2$. By definition of $L_2$ it must be $(p + m + 1) = (p + n) + 1$ and $(p + 2) = (p + n) + 2$. This is possible only if $n = m = 0$, contradicting the fact that $v$ and $y$ are not both empty.

• $L_3 = \{a^n b^m c^n | n, m \geq 0\}$ **Solution:** The language is context free because it is generated by the following grammar:

$$S \rightarrow Sc | T$$

$$T \rightarrow aTb | \epsilon$$

• $L_4 = L_1 \cap L_3 = \{a^n b^m c^{2m} | n \geq 0\}$ **Solution:** We prove that the language is not context free using the pumping lemma for context free languages. The proof is similar to the one for $L_2$. Assume for contradiction that $L_4$ is context free and let $p$ be the pumping length. The string $w = a^p b^p c^{2p}$ is in $L_4$ and has length bigger than $p$. Therefore, by the pumping lemma there exist substrings $wx_iz = w$ such that $v$ and $y$ are not both empty and $w^i xy^iz \in L_4$ for all $i$. As in the proof for $L_4$ the substrings $v$ and $y$ must be of the form $s^k$ where $s$ is one of $a,b,c$. So, assume that $v = a^n$ and $y = b^m$ (the other cases are similar). Then the string $w^2 xy^2 z = a^{p+n} b^{p+m} c^{2p}$ belongs to $L_4$. By definition of $L_4$ it must be $2p = 2(p + n) = 2(p + m)$. This is possible only if $n = m = 0$, contradicting the fact that $v$ and $y$ are not both empty.

Notice that although $L_4$ is the intersection of two context free languages, $L_4$ is not context free. This proves that context free languages are not closed under intersection, i.e., the intersection of context free languages is not necessarily context free. From this observation it also follows that context free languages are not closed under complement, i.e., the complement of a context free language is not necessarily context free.