[Problem 1.]

[Linz, Section 11.1, Exercise 10]

Yes. Suppose $L_1$ and $L_2$ are recursive languages and can be accepted by turing machines $M_1$ and $M_2$, respectively. We construct a turing machine $\hat{M}$, such that:

1. Given an input $w \in \Sigma^*$
2. Break $w$ into two substrings $w_1$ and $w_2$
   
   (a) Run $M_1$ with $w_1$ and $M_2$ with $w_2$ separately in parallel
   (b) If $w_1 \in L(M_1)$ and $w_2 \in L(M_2)$, then $\hat{M}$ accepts $w$.

Clearly, $\hat{M}$ accepts $L_1 \cdot L_2$. Also, $\hat{M}$ can decide $w \in \Sigma^*$ in finite number of steps, because step 2 takes at most $|w| - 1$ iterations, and step 2(a) can be done in finite steps ($L_1$ and $L_2$ are recursive). Therefore, $L_1 \cdot L_2$ is recursive.

[Linz, Section 11.1, Exercise 16]

Proof by contradiction. Suppose $S_1 - S_2$ is finite, and therefore is countable. Then $S_2$ must countable, since $S_1$ is countable. However, $S_2$ is uncountable, thus $S_2 - S_1$ is infinite and uncountable.

[Linz, Section 11.1, Exercise 19]

Consider numbers between 0 and 1. Irrational numbers have been defined as decimal (non-periodic) fractions. Assume it is possible to enumerate all such decimals. Let’s choose an enumeration and list the decimals in the corresponding order:

\[ a_1 = 0.a_{11}a_{12}a_{13}a_{14}... \]
\[ a_2 = 0.a_{21}a_{22}a_{23}a_{24}... \]
\[ a_3 = 0.a_{31}a_{32}a_{33}a_{34}... \]
...

where $a_{mn}$ stands for the $n^{th}$ digit of the $m^{th}$ decimal. Apply Cantor’s diagonal process. To remind, we made an assumption that all the decimals between 0 and 1 have been listed in the
sequence above. Proof by contradiction by showing that at least one decimal is missing from the list. The decimal \( b = 0.b_1b_2... \) is constructed a digit by digit. Select \( b_1 \) to be any digit but \( a_{11} \). Select \( b_2 \) to be any digit but \( a_{22} \). And in general, select \( b_n \) to be any digit but \( a_{nn} \). Then \( b \) can’t equal any decimal \( a_n, n = 1, 2, 3, ... \) because \( b \) differs from \( a_1 \) in the first digit; it differs from \( a_2 \) in the second digit and so on.

[Linz, Section 11.2, Exercise 8]

For each production rule \( u \to v \), where \( |u|, |v| > 2 \), rewrite \( u \) and \( v \) such that \( |u|, |v| \leq 2 \) and \( |u| \leq v \). For example, let us rewrite \( aABcC \to aBAc \):

The first step is to rewrite \( u \):

\[
\begin{align*}
aB & \to V_1V_0 \\
V_1c & \to V_2V_0 \\
V_2C & \to aBAc \\
V_0 & \to \lambda
\end{align*}
\]

The second step is to rewrite \( v \) using similar techniques that convert CFGs to Chomsky normal form:

\[
\begin{align*}
V_2 & \to V_3V_4 \\
V_3 & \to aB \\
V_4 & \to Ac
\end{align*}
\]

For \( u \to v \), where \( |u| = 2 \) and \( |v| = 1 \), append \( V_0 \) to \( v \) as used in Exercise 7.

[Problem 2.]

[a.]
[b.]

\[
\begin{align*}
T &\rightarrow aATZ|bBTZ|aa|bb \\
Aa &\rightarrow aA \\
Ab &\rightarrow bA \\
Ba &\rightarrow aB \\
Bb &\rightarrow bB \\
AZ &\rightarrow Za \\
BZ &\rightarrow Zb \\
Z &\rightarrow \lambda
\end{align*}
\]

Z is used to prevent subsequent swapping.