1. (60 points) Give a context-free grammar for each of the following languages. Explain how the grammar works. No points will be given if the CFG is not commented.

(a) \( L = \emptyset \)

(b) \( L = \{\lambda\} \)

(c) \( L = \{w \mid w \in \{a, b, c\}^*, w \text{ does not contain substring } bc\} = c^*(b \cup ac^*)^* \)

(d) ROLL is the language generated by the following recursive definition:

- **basis**: “single-stroke:” \( \in \) ROLL, “double-stroke:” \( \in \) ROLL, “silly-stroke:” \( \in \) ROLL.

- **recursive step**: If \( w \in \text{ROLL} \) and \( w \) contains “single” then \( wRL \) is in ROLL.
  If \( w \in \text{ROLL} \) and \( w \) contains “double” then \( wRRLL \) is in ROLL.
  If \( w \in \text{ROLL} \) and \( w = xy \) where \( x = \) “silly-stroke:”) then \( xRyL \) is in ROLL.

- **closure**: A string \( w \in \text{ROLL} \) only if it can be obtained from the basis set by a finite number of applications of the recursive step.

(e) \( L = \{a^n b^m c^{2n+m} \mid n, m \geq 0\} \)

(f) \( L = \{a^n b^m \mid n \neq m, n \geq 0, m \geq 0\} \)
   (Hint: “not equal to” means “less than or greater than”.)

2. (10 + 5 points) Consider the following grammar:

\[
S \rightarrow aSA | \lambda \\
A \rightarrow bA | \lambda
\]

(a) Prove the grammar is ambiguous by finding a string that has two distinct leftmost derivations. Show the two derivations.

(b) Build the derivation trees for the derivations in part (a).

Please turn the page over.
3. (10 + 5 + 10 points) Consider the following grammar \( G \) over \( \Sigma = \{a, b\} \).

\[
\begin{align*}
S & \rightarrow AB \mid aB \\
A & \rightarrow aB \mid BS \\
B & \rightarrow bB \mid b \mid \lambda
\end{align*}
\]

(a) Transform the grammar into \( G_1 \) so that the recursion to the start symbol is removed.

(b) Show the set of nullable variables in the new grammar \( G_1 \).

(c) Construct an essentially noncontracting grammar \( G_L \) (with a non-recursive start symbol) equivalent to \( G_1 \). An “essentially noncontracting grammar” is a grammar that has no \( \lambda \)-rules.