Question 1  Consider the following parse tree:

Now assume that we add head-words to the non-terminals in the parse tree. We do this by specifying the following rules for finding the heads of context-free rules (note that these rules don’t necessarily make sense from a linguistic standpoint):

- For the rule \( S \rightarrow NP\ VP \), the \( VP \) is the head of the rule.
- For the rule \( NP \rightarrow D\ N \), the \( N \) is the head of the rule.
- For the rule \( PP \rightarrow P\ NP \), the \( NP \) is the head of the rule.
- For the rule \( VP \rightarrow V\ NP\ PP \), the \( NP \) is the head of the rule.
- As is usual with head-finding rules, for any rule of the form \( X \rightarrow w \) where \( X \) is a non-terminal, and \( w \) is a word, we take \( w \) to be the head of the rule (and \( X \) then has \( w \) as its head-word).

Question: Draw a new version of the parse tree, where head-words have been added to the non-terminals in the tree using the rules we have specified.
**Question 2**  Consider the following lexicalized grammar:

\[
\begin{align*}
S(\text{likes}) \rightarrow & \ 2 \ NP(\text{Bob}) \ VP(\text{likes}) \\
VP(\text{likes}) \rightarrow & \ 1 \ VB(\text{likes}) \ NP(\text{parks}) \\
NP(\text{parks}) \rightarrow & \ 1 \ NP(\text{parks}) \ PP(\text{in}) \\
NP(\text{parks}) \rightarrow & \ 1 \ NP(\text{parks}) \ NP-CC(\text{London}) \\
NP(\text{Paris}) \rightarrow & \ 1 \ NP(\text{Paris}) \ NP-CC(\text{London}) \\
PP(\text{in}) \rightarrow & \ 1 \ IN(\text{in}) \ NP(\text{Paris}) \\
NP-CC(\text{London}) \rightarrow & \ 2 \ CC(\text{and}) \ NP(\text{London}) \\
NP(\text{Bob}) \rightarrow & \ Bob \\
NP(\text{parks}) \rightarrow & \ parks \\
NP(\text{London}) \rightarrow & \ London \\
NP(\text{Paris}) \rightarrow & \ Paris \\
VB(\text{likes}) \rightarrow & \ likes
\end{align*}
\]

Show all parse trees under this grammar for the sentence

*Bob likes parks in Paris and London*

If we add probabilities to the above rules, how will this model perform compare to a conventional PCFG with rules \( S \rightarrow NP \ VP \), \( NP \rightarrow NP \ PP \) etc.?
Question 3  We define the following type of “lexicalized” grammar:

- \( N \) is a set of non-terminal symbols
- \( \Sigma \) is a set of terminal symbols
- \( R \) is a set of rules which take one of two forms:
  - \( X(h) \rightarrow Y_1(h) Y_2(w) \) for \( X \in N \), and \( Y_1, Y_2 \in N \), and \( h, w \in \Sigma \)
  - \( X(h) \rightarrow h \) for \( X \in N \), and \( h \in \Sigma \)
- \( S \in N \) is a distinguished start symbol

Note that this is similar to the “lexicalized Chomsky normal form” grammar we introduced in lecture, except that we do not allow rules of the following form:

\[ X(h) \rightarrow Y_1(w) Y_2(h) \] for \( X \in N \), and \( Y_1, Y_2 \in N \), and \( h, w \in \Sigma \).

Question: Define a grammar in the above form that gives at least one valid parse tree for the sentence *the man saw the man with the man*. Draw a parse tree under your grammar for this sentence. Make sure to show the head words in your parse tree.

Question: Now assume we have a probabilistic lexicalized context-free grammar with rules that take the above form. Give pseudo-code for an efficient dynamic programming algorithm that returns the highest probability parse tree for a given input sentence \( w_1, w_2, \ldots, w_n \). Your algorithm should run in at most \( O(n^3 |N|^3) \) time where \( n \) is the length of the sentence, and \( |N| \) is the number of non-terminals in the grammar.
Question 4  Consider a language model with a vocabulary $V = \{a, b\}$ and the following parameters:

\[
\begin{align*}
q(a|\ast) &= 0.3 \\
q(b|\ast) &= 0.7 \\
q(a|a) &= 0.2 \\
q(b|a) &= 0.7 \\
q(STOP|a) &= 0.1 \\
q(a|b) &= 0.8 \\
q(b|b) &= 0.1 \\
q(STOP|b) &= 0.1
\end{align*}
\]

Now write down a lexicalized PCFG with the same distribution over sentences as this grammar (write down both the rules in the lexicalized PCFG, and their probabilities). For any string $s$, if $T(s)$ is the set of possible parse trees for $s$, and $p(t)$ is the probability for any tree, then the probability distribution over strings is

\[
p(s) = \sum_{t \in T(s)} p(t)
\]

Note that you can assume that the strings $s$ always end in the STOP symbol, so for example the string

\[
a \ b \ STOP
\]

should have probability

\[
q(a|\ast) \times q(b|a) \times q(STOP|b)
\]

under the lexicalized PCFG.