1. Consider the following context-free grammar:

\[
\begin{align*}
\text{ROOT} & \rightarrow \text{yeah S} && \text{Det} \rightarrow \text{the} \\
\text{S} & \rightarrow \text{NP VP} && \text{N} \rightarrow \text{girl} \\
\text{NP} & \rightarrow \text{Det N} && \text{N} \rightarrow \text{glasses} \\
\text{NP} & \rightarrow \text{Det Adj N} && \text{Adj} \rightarrow \text{tall} \\
\text{NP} & \rightarrow \text{N} && \text{V} \rightarrow \text{put} \\
\text{NP} & \rightarrow \text{NP PP} && \text{Part} \rightarrow \text{out} \\
\text{VP} & \rightarrow \text{V Part NP} && \text{N} \rightarrow \text{cat} \\
\text{VP} & \rightarrow \text{V} && \text{P} \rightarrow \text{with} \\
\text{PP} & \rightarrow \text{P NP} & \\
\end{align*}
\]

and the following sentence:

(1) yeah the tall girl with glasses put out the cat

2. Chomsky Normal Form: This is a grammar in which each rule takes one of the three following forms:

- \( \text{ROOT} \rightarrow \epsilon \) where \( \text{ROOT} \) is the start symbol;
- \( X \rightarrow Y Z \) where \( Y \) and \( Z \) are non-terminals; or
- \( X \rightarrow t \) where \( t \) is a terminal.

This form facilitates parsing.

3. Converting to Chomsky Normal Form: There are several strategies we can use for conversion:

(a) To break up rules that have more than two right-hand symbols, we can introduce “intermediate” non-terminals to break the rules up into simpler components. e.g., convert

\[ X \rightarrow Y_1 Y_2 \ldots Y_n \]
into

\[ X \rightarrow Y_1 X' \]
\[ X' \rightarrow Y_2 \ldots Y_n \]

where \( X' \) is a new nonterminal symbol that doesn’t already appear elsewhere in the grammar.

(b) To break up rules that mix non-terminals and terminals on the right-hand side, we can likewise introduce “intermediate” non-terminals. e.g., convert

\[ X \rightarrow Y_1 \ldots Y_{i-1} t Y_{i+1} \ldots Y_n \]

where \( t \) is a terminal into

\[ X \rightarrow Y_1 \ldots Y_{i-1} X' Y_{i+1} \ldots Y_n \]
\[ X' \rightarrow t \]

(c) To deal with rules that have only a single non-terminal on the right-hand side (UNIT PRODUCTIONS), we use the concept of UNARY CLOSURE. If a sequence of unit productions takes category \( X \) to category \( Y \), then for every rule \( Y \rightarrow \alpha \) in the grammar, add a rule \( X \rightarrow \alpha \). Once that’s all done, remove all the unary productions.

4. The Cocke-Kasami-Younger (CKY, or CYK) Parsing Algorithm is a bottom-up parser with dynamic programming. For a CFG in Chomsky Normal Form, here is the CKY algorithm:

**Algorithm 1 CKY Parsing**

1: function CKY-PARSE(\( \text{words,grammar} \))
2: Initialize table to the upper half of an \( n \times n \) matrix
3: for \( k \) in 1 to len(\( \text{words} \)) do
4: \hspace{1em} for \( i \) in \( k-1 \) to 0, incrementing in step size \(-1\) do
5: \hspace{2em} if \( i == k-1 \) then
6: \hspace{3em} for rule \( X \rightarrow t \) such that \( \text{words}[k]==t \) do
7: \hspace{4em} Put \( X \) in \( \text{table}[i,k] \)
8: \hspace{2em} else
9: \hspace{3em} for \( j \) in \( i+1 \) to \( j-1 \) do
10: \hspace{4em} for rule \( X \rightarrow Y \ Z \) such that \( Y \in \text{table}[i,j] \) and \( Z \in \text{table}[j,k] \) do
11: \hspace{5em} Put \( X \) in \( \text{table}[i,k] \)
12: return table

5. Now let’s work out CKY parsing of sentence [1]. We’ll need to (i) convert the grammar to Chomsky Normal Form, and then (ii) run the algorithm.