Exercises on Grammars

1. Consider the following context-sensitive grammar.

$$\begin{aligned} X &\rightarrow \texttt{a}Y\texttt{b} \\ \texttt{a}Y &\rightarrow \texttt{ab}Y \\ Y\texttt{b} &\rightarrow \texttt{ab} \end{aligned}$$

   Describe the language generated by this grammar, assuming X is the start symbol.

2. Consider the following context-sensitive grammar.

$$\begin{aligned} X &\rightarrow \texttt{a}Y\texttt{b} \\ \texttt{a}Y &\rightarrow \texttt{ba}Y \\ Y\texttt{b} &\rightarrow \texttt{ab} \end{aligned}$$

   Describe the language generated by this grammar, assuming X is the start symbol.

3. The following grammar describes C-style expressions that involve array elements and addition. Assume that **X** is a terminal that stands for a variable identifier and **N** is a terminal that stands for a numeric constant. The other terminals are `[`, `]`, and `+`.

$$E \quad ::= \quad \mathbf{X}\texttt{[}E\texttt{]} \mid E\texttt{+}E \mid \mathbf{X} \mid \mathbf{N}$$

   An example of an expression generated by this grammar is `a[i+2]+b`, assuming `a` and `b` are in **X** and `2` is in **N**.

   (a) Give an example of an expression generated by this grammar that admits two different parse trees, and thus demonstrates that the grammar is ambiguous.

   (b) Resolve the ambiguity in the grammar, being sure to follow the standard rules of associativity and precedence for the standard arithmetic operators.

   (c) Describe the $NULLABLE$, $FIRST$, and $FOLLOW$ sets for each non-terminal in your grammar.[1]

[0] I'd like to thank Maia Ramambason for finding and fixing some errors in an earlier version of this document.

[1] Note that some authors avoid the $NULLABLE$ set. Instead, to indicate that a non-terminal X can generate the empty word, they put ϵ into $FIRST(X)$. Personally, I find this a bit of a hack, so I prefer to keep the $FIRST(X)$ set for the symbols that can actually begin a word generated from X, and separately to use $NULLABLE(X)$ to record whether X can generate the empty word.

(d) Explain why your grammar is not suited to top-down parsing.

(e) Rewrite your grammar so that it becomes suited to top-down parsing.

(f) Describe the $NULLABLE$, $FIRST$, and $FOLLOW$ sets for each non-terminal in your new grammar.

(g) Fill in the following predictive parsing table with a row for each non-terminal and a column for each terminal (including $ for the end-of-input symbol). For each cell in row X and column t, identify which production rule should be used if the parser is seeking to parse the non-terminal X and the current terminal is t.

	+	[	]	**X**	**N**	$
E						
⋮						

Do you find that your parser does not know what to do when it meets an **X**? This problem can either be fixed by using an extra token of lookahead, which complicates your parsing table with an extra dimension, or by just rewriting your grammar (yet again!).

(h) We shall now turn our attention to bottom-up (shift/reduce) parsing. Return to the unambiguous grammar that you produced in part (b), and add a new start symbol, $S \to E$.

i. Write out the initial item-set, which is obtained by taking the closure of the item $S \to \bullet E$.

ii. Write out the rest of the item-sets, labelling the transitions between them with the terminals and non-terminals of your grammar.

iii. Fill in the following shift/reduce parsing table, with one 'action' column for each terminal (including $), one 'goto' column for each non-terminal, and one row for each item-set.

	action						*goto*				
	+	[	]	**X**	**N**	$	S	E	E'	T	T'
0											
1											
⋮											