Exercise 4.4.5. The grammar $S \to aSa \mid aa$ generates all even-length strings of a's. We can devise a recursive-descent parser with backtrack for this grammar. If we choose to expand by production $S \to aa$ first, then we shall only recognize the string *aa.* Thus, any reasonable recursive-descent parser will try $S \to aSa$ first.

a) Show that this recursive-descent parser recognizes inputs *aa, aaaa,* and *aaaaaaaa,* but not *aaaaaa.*
b) What language does this recursive-descent parser recognize?

The following exercises are useful steps in the construction of a "Chomsky Normal Form" grammar from arbitrary grammars, as defined in Exercise 4.4.8.

Exercise 4.4.6. A grammar is *ε-free* if no production body is ε (called an *ε-productiori).*
a) Give an algorithm to convert any grammar into an e-free grammar that generates the same language (with the possible exception of the empty string — no e-free grammar can generate e).
b) Apply your algorithm to the grammar *S aSbS \ bSaS* | ε. *Hint:* First find all the nonterminals that are *nullable,* meaning that they generate ε, perhaps by a long derivation.

Exercise 4.4.7. A *single production* is a production whose body is a single nonterminal, i.e., a production of the form $A \to B$.
a) Give an algorithm to convert any grammar into an ε-free grammar, with no single productions, that generates the same language (with the possible exception of the empty string) *Hint:* First eliminate ε-productions, and then find for which pairs of nonterminals *A* and *B* does $A \to B$ by a sequence of single productions.
b) Apply your algorithm to the grammar
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \mathrm{id}$
c) Show that, as a consequence of part (a), we can convert a grammar into an equivalent grammar that has no *cycles* (derivations of one or more steps in which $A \to A$ for some nonterminal *A).*

Exercise 4.4.8: A grammar is said to be in *Chomsky Normal Form* (CNF) if every production is either of the form $A \to BC$ or of the form $A \to a$, where *A*, *B*, and *C* are nonterminals, and *a* is a terminal. Show how to convert any grammar into a CNF grammar for the same language (with the possible exception of the empty string — no CNF grammar can generate ε).

Exercise 4.4.9. Every language that has a context-free grammar can be recognized in at most $O(n^3)$ time for strings of length *n*. A simple way to do so, called the *Cocke-Younger-Kasami* (or CYK) algorithm is based on dynamic programming. That is, given a string $a_1a_2...a_n$ we construct an n-by-n table *T* such that T_{ij} is the set of nonterminals that generate the substring $a_ia_{i+1}...a_j$.If the underlying grammar is in CNF (see Exercise 4.4.8), then one table entry can be filled in in $O(n)$ time, provided we fill the entries in the proper order: lowest value of j-i first. Write an algorithm that correctly fills in the entries of the table.