Storing CFGs and `hasLambdaRule(`X`)`

You've now worked through several exercises centered around **context free grammars**. We are embarking on the topic *grammar analysis*, so now is a good time to consider CFGs from an implementation perspective.

Discuss within your learning groups: **what type of data structure(s) or object(s) would you use to store a CFG in computer memory**?

How would you represent that A has a rewrite rule of $A \rightarrow \lambda$? Is this implicit in the data of the structure, or would you use a member (helper) function?

CFGs in Memory & `hasLambdaRule(`X`)`

- Not all grammars use S as the **goal** or **starting symbol**, so you'll definitely want some data member `.goal` or `.startsymbol`.
- A CFG is mostly a collection of **rewrite** or **production** rules, and non-terminals can have more than one RHS. So an *associative array of lists* or a "multiset" seems most applicable (`.rules`?).
- You could maintain a separate set of all non-terminals that have λ as a RHS,
- or, **since RHS are independent of each other, with no required or implicit ordering** you could keep the RHS lists ("values" of `.rules`) sorted by ascending length, and have a simple query function:

  procedure `hasLambdaRule(theCFG, X)` $\equiv$ $|$`theCFG.rules[X][0]`$| = 0$

While there are plenty wrong ways to represent a CFG, there is certainly **more than one "right way,"** something you should be thinking about . . .