CKY, RECOGNIZER VERSION.

INPUT: string of n words.
OUTPUT: yes / no
DATA STRUCTURE: $n \times n$ table.
rows labeled 0 to $n-1$
columns labeled 1 to n
cell (i, j) lists constituents found between i and j

For each i from 1 to n:
    Add to $(i-1, i)$ all categories allowed for the word between $i-1$ and i

For width from 2 to n:
    For start from 0 to n - width:
        Define end to be start + width
        For mid from start+1 to end-1
            For every constituent in (start, mid)
                For every constit. in (mid end)
                    For all ways of combining them (if any):
                        Add the resulting constit (start, end) if it's not already there.

EARLEY'S ALGORITHM. (1970)

Nice combo of our previous ideas from today:

- incremental interpretation
- no restrictions on the form of the grammar
  (A → B C spoon D x is an okay rule
  thanks to dotted rules)
- $O(n^3)$ worst case, but faster for many grammars
- uses left context and optionally right context to constrain search

---

INPUT: string of n words

OUTPUT: yes/no (i.e., recognizer, but can turn into parser)

DATA STRUCTURE: columns 0 thru n,
corresponding to gaps between words
column j is a list of entries like
(i, A → X Y . Z W)
meaning there could be an A
starting at i, and we have
found the X Y part of it
from i to j.

EARLEY'S ALG, RECOGNIZER VERSION, NO LOOKAHEAD.

Add ROOT → .S to column 0.

For each j from 0 to n:

- For each dotted rule in column j, (including those we add as we go!), look at what's after the dot:
  - If it's a word w, SCAN:
    - If w matches the input word between j and j+1, advance the dot and add the resulting rule to column j+1.
  - If it's a nonterminal X, PREDICT:
    - Add all rules for X to the bottom of column j, with the dot at the start: e.g., X → .Y Z
  - If there's nothing after the dot, COMPLETE:
    - We've finished some constituent A that started in column i < j. So for each rule in column i that has A after the dot:
      - Advance the dot and add the result to the bottom of column j.

Output "yes" just if we have ROOT → S. in column n.

NOTE: Don't add an entry to a column if it's already there!