1) \( \text{Predict}( s \rightarrow w) = \{a\} \)

2) \( \text{Predict}( s \rightarrow s \text{tail1}) = \{\epsilon\} \)

3) \( \text{Predict}( \text{tail1} \rightarrow \epsilon) = \{\epsilon\} \)

4) \( \text{Predict}( \text{tail1} \rightarrow s) = \{s, a\} \)

5) \( \text{Predict}( w \rightarrow a \text{tail1}) = \{a\} \)

6) \( \text{Predict}( \text{tail1} \rightarrow bb) = \{bb\} \)

7) \( \text{Predict}( \text{tail1} \rightarrow \text{tail2}) = \{a\} \)

---

push start sym on stack

loop

if top is terminal

if top = EOI then done!
else if next = top
    throw away
else error
else if table[top, next] != Null
    push rhs (reverse)
else error
stmt → if cond then- clause else- clause
then- clause → stmt
else- clause → else stmt | ε

if (true)
  if (x > 0)
    stmt
  else
    y = x;
  end-if
  stmt
  y = x

if (true)
  stmt
  y = x

if true else- clause
  stmt
  y = x

soln #1: modify parser to choose rule (1st listed, perhaps)
soln #2: modify syntax
**Bottom-up parsing**

key idea: keep track of how far along you are in rules

LR item: rule w/ marker indicating progress

\[ S \rightarrow \epsilon \ & \ \text{fail} \]
\[ S \rightarrow \omega \w \]

\[ w \rightarrow \alpha \ \text{fail}_2 \]

- closure: all rules with LHS = non-terminal that precedes

- state: set of items and their closures

- stack: stack to keep track of previous states

actions: shift - adds a symbol and state to stack
reduce - combine rhs of a rule into its lhs
shift/reduce conflict: take either at end of rhs but also in middle

LR(0): no shift/reduce conflicts  (if have end-of-input marker, any deterministic bottom-up parseable any less an LR(0) grammar) is probably very ugly