Concepts of Programming Languages, CSCI 305, Fall 2021
Recursive Descent Parsing – Table Driven, Oct. 11

Table Driven Recursive Descent parser code in Figure 2.19, page 83.

- Code is independent of the language, the table is language dependent.
- Parsing tables are typically produced by an automatic tool.

Note the similarity of this table with the scanner table (for table-driven scanning) in Figure 2.11, page 66.
Remember, for that one we had (page 67):

```
scan_tab: array[char, state] of record
        action: (move, recognize, error)
        new_state: state
```

This time we have (page 84):

```
parse_tab: array[non-terminal, terminal] of record
        action: (predict, error)
        prod: production
```

The tricky part to creating a recursive descent parser is figuring out which tokens should label the arms of the case statements. There are situations when a token X needs to be placed into the arm of a case statement:

1. The right-hand side of the production may yield a string beginning with X (FIRST)
2. The right-hand side may yield nothing and X begins the yield of what comes next (FOLLOW)