Concepts of Programming Languages, CSCI 305, Fall 2021
Recursive Descent Parsing – Subroutines, pages 73-78, Oct. 6

Class Exercise - Parsing using subroutines , Parsing using a table

program ⟶ *stmt_list* `$$`
stmt_list ⟶ *stmt stmt_list* | ϵ
stmt ⟶ `id :=` *expr* | `read id` | `write` *expr*
expr ⟶ *term term_tail*
term_tail ⟶ *add_op term term_tail* | ϵ
term ⟶ *factor factor_tail*
factor_tail ⟶ *mult_op factor factor_tail* | ϵ
factor ⟶ `(` *expr* `)` | `id` | `number`
add_op ⟶ `+` | `-`
mult_op ⟶ `*` | `/`

```
procedure match(expected)
    if input_token = expected then consume_input_token()
    else parse_error

-- this is the start routine:
procedure program()
    case input_token of
        id, read, write, $$ :
            stmt_list()
            match($$)
        otherwise parse_error

procedure stmt_list()
    case input_token of
        id, read, write : stmt(); stmt_list()
        $$ : skip       -- epsilon production
        otherwise parse_error

procedure stmt()
    case input_token of
        id : match(id); match(:=); expr()
        read : match(read); match(id)
        write : match(write); expr()
        otherwise parse_error

procedure expr()
    case input_token of
        id, number, ( : term(); term_tail()
        otherwise parse_error

procedure term_tail()
    case input_token of
        +, - : add_op(); term(); term_tail()
        ), id, read, write, $$ :
            skip       -- epsilon production
        otherwise parse_error

procedure term()
    case input_token of
        id, number, ( : factor(); factor_tail()
        otherwise parse_error
```

```
procedure factor_tail()
    case input_token of
        *, / : mult_op(); factor(); factor_tail()
        +, -, ), id, read, write, $$ :
            skip        -- epsilon production
        otherwise parse_error

procedure factor()
    case input_token of
        id : match(id)
        number : match(number)
        ( : match(); expr(); match())
        otherwise parse_error

procedure add_op()
    case input_token of
        + : match(+)
        - : match(-)
        otherwise parse_error

procedure mult_op()
    case input_token of
        * : match(*)
        / : match(/)
        otherwise parse_error
```

Hand-execute the recursive descent parsing code that uses subroutines on the program:

read A
read B
sum := A + B
write sum
write sum / 2
$$ is sent to indicate the end-of-file'

input_token is 'read' initially, later 'id', later 'read', …

```
call program(){
    call stmt_list(){
        call stmt(){
            match(read) --- input_token becomes id(A)
            match(id)  ---- input_token becomes read
        }
        call stmt_list(){
            call stmt(){
                match(read) ---- input_token becomes id(B)
                match(id) -----input_token becomes id(sum)
            }
            call stmt_list(){
                call stmt(){
                    match(id) --- input_token becomes :=
                    match(:=)  --- inptput_token becomes id(A)
                    call expr() {
                        call term(){
                            call factor(){
```

```
                                                                match(id) --- +
                                                        }
                                                        call factor_tail() {
                                                        }
                                                }
                                                call term_tail(){
                                                        call add_op(){
                                                                match(+) ---- B
                                                        }
                                                        call term(){
                                                                call factor(){
                                                                        match(id) --write
                                                                }
                                                                call factor_tail(){
                                                                }
                                                        }
                                                        call term_tail(){
                                                        }
                                                }
                                        }
                                }
                                call stmt_list(){
                                        call stmt(){
                                                match(write) ---sum
                                                call expr(){
                                                        call term(){
                                                                call factor(){
                                                                        match(id) -- write
                                                                }
                                                                call factor_tail(){
                                                                }
                                                        }
                                                        call term_tail(){
                                                        }
                                                }
                                        }
                                        call stmt_list(){
                                                call stmt(){
                                                        match(write) ---- sum
                                                        call expr(){
                                                                …  ---- $$
                                                        }
                                                }
                                                call stmt_list(){
                                                }
                                        }

                                }
                        }
                }
        }
        match($$)
}
```

Draw the parse tree, and see how the order in which the subroutines are called matches the parse tree.
Recursive descent parsing using tables.