Favor Wariboko

Assign 6

1.

the grammar above is ambigous and this line for example is the reason: <expr>::= <expr> ; <expr> . and <expr> = let <expr> in <expr>

With a line like ↑ if we plug in an example like <exp>::= let x in y ; z , it can be parsed in two ways.

1. (let x in y); z
2. (let x in (y; z)

Because of this it is ambigous

2.

<id> ::= a | b | c | ... | z

<dig> ::= 0 | 1 | 2 | . . . | 9

<expr> ::= <simple_expr> | <let_expr> | <seq_expr> | <beg-expr>

<simple_expr> ::= () | (dig> | <id>

<let_expr> ::= let (id) = <expr> in <expr>

<seq_expr> ::= <expr> ; <expr>

<begin_expr> ::= begin <expr> end

3. <the simple_expr> - handles the basic cases without Ambigui such as empty parenthesis, digits, and identifiers

<Let expression>: clearly define the syntax for let expressions, and it removing ambiguity in the declaration and usage of variables

sequence_expr: Introduces a clear structure for sequences, ensuring that expressions are properly seperated by semicolons

Begin_expr: provide a distinct syntax for expression enclosed in `begin` and `end`, avoiding confusion with constructs. This would make it more clear and less ambiguous