<program> → <block> '.'

<block> → <declaration> <statement>

<declaration> → <constant\_decl> <var\_decl> <proc\_decl>

<const-decl> → 'const' <const-list> ';'

<const-decl> → ε

<const-list> → [ident] '=' <number-or-string> <const-A>

<const-A> → ',' [ident] '=' <number-or-string> <const-A>

<const-A> → ε

<var-decl> → 'var' <ident-list> ':' <type> ';'

<var-decl> → ε

<ident-list> → [ident] <ident-A>

<ident-A> → ',' [ident] <ident-A>

<ident-A> → ε

<proc-decl> → ε <proc-A>

<proc-A> → 'procedure' [ident] ';' <block> ';' <proc-A>

<proc-A> → ε

<type> → 'integer' | 'string' | 'boolean'

<statement> → <assign-statement> | <call-statement> | <begin-statement>|

<if-statement> | <while-statment> | <print-statement>|

<read-statement> | ε

<assign-statement> → [ident] ':=' <expression>

<call-statement> → 'call' [ident]

<begin-statement> → 'begin' <statement-list> 'end'

<if-statement> → 'if' <condition> 'then' <statement>

<while-statment> → 'while' <condition> 'do' <statement>

<print-statement> → 'print' [ident]

<read-statement> → 'read' [ident]

<statement-list> → <statement> <statement-A>

<statement-A> → ';' <statement> <statement-A>

<statement-A> → ε

<condition> → 'odd' <expression>

<condition> → <expression> <relop> <expression>

<expression> → <term> <expression-A>

<expression-A> → <add-subop> <term> <expression-A>

<expression-A> → ε

<term> → <factor> <term-A>

<term-A> → <mult-divop> <factor> <term-A>

<term-A> → ε

<factor> → [ident]

<factor> → [number]

<factor> → '(' <expression> ')'

<factor> → <string\_literal>

<number-or-string> → [number] | <string\_literal>

<string\_literal> → '”' [A-Za-z0-9\_]{0, 30} '”'

<add-subop> → '+' | '-'

<mult-divop> → '\*' | '\'

<relop> → '=' | '<>' | '<' | '>' | '<=' | '>='