Appendix A

PL Detective, Syntax

<Program>	→	<Block> | <Block> ;
<DeclList>	→	<Decl> | <Decl> ; <DeclList> | ε
<Decl>	→	VAR id : <Type> | TYPE id = <Type> | <ProcDecl>
<ProcDecl>	→	PROCEDURE id (<Formals>) : <Type> = <Block> | PROCEDURE id (<Formals>) = <Block>
<Formals>	→	<FormalList> | ε
<FormalList>	→	<Formal> | <FormalList> ; <Formal>
<Formal>	→	id : <Type>
<Type>	→	INTEGER | <SubrTy> | <ArrayTy> | id | <ProcTy>
<SubrTy>	→	[Number TO Number]
<ArrayTy>	→	ARRAY <SubrTy> OF <Type>
<ProcTy>	→	PROCEDURE (<Formals>) : <Type> | PROCEDURE (<Formals>)
<Block>	→	<DeclList> BEGIN <StmtList> END
<StmtList>	→	<Stmt> | <Stmt> ; <StmtList> | ε
<Stmt>	→	<Assignment> | <Return> | <Block> | <Conditional> | <Iteration> | <Output> | <Expr>
<Assignment>	→	<Expr> := <Expr>
<Return>	→	RETURN <Expr>
<Conditional>	→	IF <Expr> THEN <StmtList> ELSE <StmtList> END
<Iteration>	→	WHILE <Expr> DO <StmtList> END
<Output>	→	PRINT <Expr>
<Expr>	→	<Operand> | <Expr> <Operator> <Operand>
<Operand>	→	Number | id | <Operand> [<Expr>] | <Operand>(<Actuals>) | (<Expr>)
<Operator>	→	+ | > | AND
<Actuals>	→	<ActualList> | ε
<ActualList>	→	<Expr> | <Actuals> , <Expr>

Notes:

- <Program> is the start symbol of the grammar
- This grammar is in BNF, not in EBNF. Particularly, the '[' and ']' are terminals in the language: they do not not mean "optional" in EBNF.
- The words in upper case are reserved words of the language (e.g., PROCEDURE and AND)
- <SubrTy> is a subrange type. For example, a variable declared to be of subrange type [1 TO 10] can hold values between 1 and 10 only.
- Numbers include both negative and positive integers.
- id (which are variable or procedure names) are a string of characters (a-zA-Z)