*Syntax-based **TYPE ANALYSIS***

SPL_PROG ::= **glob** { VARIABLES }
proc { PROCDEFS }
func { FUNCDEFS }
main { MAINPROG }

Semantic Attribution:
SPL_PROG is correctly typed **if** VARIABLES is correctly typed
and if PROCDEFS is correctly typed
and if FUNCDEFS is correctly typed
and if MAINPROG is correctly typed.

VARIABLES ::= *// nothing, nullable*

Semantic Attribution: VARIABLES **is** correctly typed (fact).

VARIABLES ::= VAR VARIABLES

Semantic Attribution:
VARIABLES (left-hand-side) is correctly typed
if VAR is of type ***"numeric"***
and if VARIABLES (right-hand-side) is correctly typed

VAR ::= user-defined-name

Semantic Attribution: VAR **is** of type ***"numeric"*** (fact);
user-defined-name **is** of type ***"numeric"*** (fact);

PROCDEFS ::= *// nothing, nullable*

Semantic Attribution: PROCDEFS **is** correctly typed (fact).

PROCDEFS ::= PDEF PROCDEFS

Semantic Attribution:
PROCDEFS (left-hand-side) is correctly typed
if PDEF is correctly typed
and if PROCDEFS (right-hand-side) is correctly typed.

PDEF ::= NAME (PARAM) { BODY }

Semantic Attribution:
PDEF is correctly typed **if** NAME is *type-less* (has *no* type in the Symbol Table),
and if PARAM is correctly typed,
and if BODY is correctly typed.

FDEF ::= NAME (PARAM) { BODY **; return** ATOM }

Semantic Attribution:
FDEF is correctly typed **if** NAME is *type-less* (has *no* type in the Symbol Table),
and if PARAM is correctly typed,
and if BODY is correctly typed, **and if** ATOM is of type ***"numeric"***

FUNCDEFS ::= FDEF FUNCDEFS

Semantic Attribution:
FUNCDEFS (left-hand-side) is correctly typed
if FDEF is correctly typed
and if FUNCDEFS (right-hand-side) is correctly typed.

FUNCDEFS ::= *// nothing, nullable*

Semantic Attribution: FUNCDEFS **is** correctly typed (fact).

BODY ::= **local** { MAXTHREE } ALGO

Semantic Attribution:
BODY is correctly typed **if** MAXTHREE is correctly typed **and if** ALGO is correctly typed.

PARAM ::= MAXTHREE

Semantic Attribution:
PARAM is correctly typed **if** MAXTHREE is correctly typed.

MAXTHREE ::= *// nothing, nullable*

Semantic Attribution: MAXTHREE **is** correctly typed (fact).

MAXTHREE ::= VAR

Semantic Attribution:
MAXTHREE is correctly typed **if** VAR is of type ***"numeric"***.

MAXTHREE ::= VAR VAR

Semantic Attribution:
MAXTHREE is correctly typed **if** both VAR are of type ***"numeric"*** (each of them)

MAXTHREE ::= VAR VAR VAR

Semantic Attribution:
MAXTHREE is correctly typed **if** all three VAR are of type ***"numeric"*** (each of them)

MAINPROG ::= **var {** VARIABLES **}** ALGO

Semantic Attribution:
MAINPROG is correctly typed **if** VARIABLES is correctly typed
and if ALGO is correctly typed.

ATOM ::= VAR

Semantic Attribution: ATOM is of type ***"numeric"*** **if** VAR is of type ***"numeric"***.

ATOM ::= number

Semantic Attribution: ATOM is of type ***"numeric"*** (fact).

ALGO ::= INSTR

Semantic Attribution: ALGO is correctly typed **if** INSTR is correctly typed.

ALGO ::= INSTR **;** ALGO

Semantic Attribution: ALGO (left-hand-side) is correctly typed
if INSTR is correctly typed **and if** ALGO (right-hand-side) is correctly typed.

INSTR ::= **halt**

Semantic Attribution: INSTR **is** correctly typed (fact).

INSTR ::= **print** OUTPUT

Semantic Attribution: INSTR is correctly typed **if** OUTPUT is correctly typed.

INSTR ::= NAME (INPUT)

Semantic Attribution:
INSTR is correctly typed **if** NAME is *type-less* (has *no* type in the Symbol Table),
and if INPUT is correctly typed.

INSTR ::= ASSIGN

Semantic Attribution: INSTR is correctly typed **if** ASSIGN is correctly typed.

INSTR ::= LOOP

Semantic Attribution: INSTR is correctly typed **if** LOOP is correctly typed.

INSTR ::= BRANCH

Semantic Attribution: INSTR is correctly typed **if** BRANCH is correctly typed.

ASSIGN ::= VAR = NAME (INPUT)

Semantic Attribution:
ASSIGN is correctly typed **if** NAME is *type-less* (has *no* type in the Symbol Table),
and if INPUT is correctly typed,
and if VAR is of type ***"numeric"***.

ASSIGN ::= VAR = TERM

Semantic Attribution:
ASSIGN is correctly typed **if** TERM is of type ***"numeric"***
and if VAR is of type ***"numeric"***.

LOOP ::= **while** TERM { ALGO }

Semantic Attribution:
LOOP is correctly typed **if** TERM is of type ***"boolean"***
and if ALGO is correctly typed.

LOOP ::= **do** { ALGO } **until** TERM

Semantic Attribution:
LOOP is correctly typed **if** TERM is of type ***"boolean"***
and if ALGO is correctly typed.

BRANCH ::= **if** TERM **{** ALGO **}**

Semantic Attribution:
BRANCH is correctly typed **if** TERM is of type ***"boolean"***
and if ALGO is correctly typed.

BRANCH ::= **if** TERM **{** ALGO **}** **else {** ALGO **}**

Semantic Attribution:
BRANCH is correctly typed **if** TERM is of type ***"boolean"***
and if both ALGO are correctly typed.

OUTPUT ::= ATOM

Semantic Attribution: OUTPUT is correctly typed **if** ATOM is of type ***"numeric"***.

OUTPUT := string

Semantic Attribution: OUTPUT **is** correctly typed (fact).

INPUT ::= *// nothing, nullable*

Semantic Attribution: INPUT **is** correctly typed (fact).

INPUT ::= ATOM

Semantic Attribution: INPUT is correctly typed **if** ATOM is of type ***"numeric"***

INPUT ::= ATOM ATOM

Semantic Attribution:
INPUT is correctly typed **if** both ATOM are of type ***"numeric"*** (each of them)

INPUT ::= ATOM ATOM ATOM

Semantic Attribution:
INPUT is correctly typed **if** all three ATOM are of type ***"numeric"*** (each of them)

TERM ::= ATOM

Semantic Attribution: TERM is of type *"numeric"* **if** ATOM is of type *"numeric"*

TERM ::= **(** UNOP TERM **)**

Semantic Attribution:
TERM (left-hand-side) is of type *"numeric"* **if** UNOP is of type *"numeric"*
and if TERM (right-hand-side) is of type *"numeric"*

EXOR

TERM (left-hand-side) is of type *"boolean"* **if** UNOP is of type *"boolean"*
and if TERM (right-hand-side) is of type *"boolean"*

UNOP ::= **neg**

Semantic Attribution: UNOP **is** of type *"numeric"* (fact).

UNOP ::= **not**

Semantic Attribution: UNOP **is** of type *"boolean"* (fact).

TERM ::= **(** TERM BINOP TERM **)**

Semantic Attribution:
TERM (left-hand-side) is of type *"numeric"* **if** BINOP is of type *"numeric"*
and if both TERM (right-hand-side) are of type *"numeric"*

EXOR

TERM (left-hand-side) is of type *"boolean"* **if** BINOP is of type *"boolean"*
and if both TERM (right-hand-side) are of type *"boolean"*

EXOR

TERM (left-hand-side) is of type *"boolean"* **if** BINOP is of type *"comparison"*
and if both TERM (right-hand-side) are of type *"numeric"*

BINOP ::= **>**

Semantic Attribution: BINOP **is** of type *"comparison"* (fact).

BINOP ::= eq

Semantic Attribution: BINOP is of type *"comparison"* (fact).

BINOP ::= or

Semantic Attribution: BINOP is of type *"boolean"* (fact).

BINOP ::= and

Semantic Attribution: BINOP is of type *"boolean"* (fact).

BINOP ::= plus

Semantic Attribution: BINOP is of type *"numeric"* (fact).

BINOP ::= minus

Semantic Attribution: BINOP is of type *"numeric"* (fact).

BINOP ::= mult

Semantic Attribution: BINOP is of type *"numeric"* (fact).

BINOP ::= div

Semantic Attribution: BINOP is of type *"numeric"* (fact).

With this information you can now implement a Type-Analyser that crawls through a given SPL Syntax Tree, updates the Symbol Table, and tells the user whether (or not) the given SPL Input Program was correctly typed.

And now : HAPPY CODING :)