**The P Language Scope and Semantic Rules**

**Scope rules:**

1. Each identifier must be declared in an enclosing scope prior to its use.
2. Identifiers declared in the declaration part of the program are within the global scope.
3. Each block is a new local scope.
4. Identifiers inside an fdef are only in the scope of the block defined by the fdef. You cannot refer to them from outside the function.
5. It is not possible to declare identifiers of the same name and type in overlapping scopes.
6. There is no built-in scope in the P Language (contrary to Python etc.).

**Semantic rules:**

General Rules:

1. A valid P program consists of two parts. A declaration part (optional) and a main body (compulsory).
2. The main body of the P program can be empty.
3. P is strong, and statically typed.

Assignment Rules:

1. Assignments are legal only if the identifier on the left-hand side and the expression on the right-hand side have the same type.
2. The language is very strongly typed:
   1. It is not possible to assign an int to a float or assign a float to an int.

Arithmetic Operations Rules:

1. Arithmetic operations can only be performed on types int and float. DUH?
2. All arithmetic operations on numbers (addition, subtraction, division, power) must be performed on numbers of the same type. The exception to this rule is that floats, as well as integers, may only be raised to an integer power. An int type cannot interact with a float or vice versa. (e.g. 2 + 2.0 is not permitted)
3. A number cannot be raised to a non-integer number. (e.g. 23.5 is not permitted.)
4. All divisions are automatically converted to float.

Comparison Operations Rules:

1. Comparison operators <, <=, >, >= can only be performed on int and float expressions. (e.g. 6 < “Doughnut” is not permitted). They cannot be applied to str, list or tuple.
2. The ==, != comparison operators can also be applied to Boolean operands. They cannot be applied to str, list or tuple.
3. As with other operations, both operands must be of the same type.

Control Flow Rules:

1. The condition of if, while and repeat statements must be of type bool.

Function Rules:

1. There cannot be any function overloading nor overriding.
2. The number and type of actual parameters of functions must match the number and type of the declared formal parameters.
3. An fdef with an outType of void can contain empty return statements. That is “return;” and not “return <exp>;”.
4. An fdef with any outType other than void, must contain at least one return statement returning an expressions whose type matches that of the outType.

Sequences rules:

1. All the elements of a list must be of the same type.
2. The elements of a tuple can be of any type.