Readme: Syntactic Specifications

The source language requires the correct coding, to make sure the compiler runs correctly.

1. Structure
   The source code must be start with 'main' or 'def function' at the first line. For example:

   ```
   main
   --main body--
   def {func}
   --func body--
   ```

   Since our project contains a linker, it is NOT a must for a program to have a 'main' function. However, all the files in a project can only have at most ONE 'main' function.

   'main' MUST appear in the first line of the program, if there is one.

2. Variables
   This language only supports declaring integer variables. The format is as follows.

   ```
   {int} a
   ```

   A variable MUST be declared before use.

   Multiple variable declaration in a single line is NOT supported. If you have multiple variables to declare, you should put them in successive lines. ONE at a time.

   ALL variables declared are global variables. You can directly use the declared variables in functions.

   Variables can ONLY be declared within the main function.

3. Functions
   Function Definition:
   This language supports declaring function, with up to 2 parameters. Function definition use the keyword 'def', as shown in the following examples.

```
def {func1}
--func body--

--or--

def {func2} a
--func body--

--or--

def {func3} a,b
--func body--
```

The name of the function must be contained within the curly braces.

It is allowed to have a parameter and a variable having the same name. If this happens, the parameter has the priority, so that calling the name only affects the parameter with the name.

The compiler will automatically detect the end of a function definition, so there is no symbol indicating the end.

NO variables should be declared within a function scope. Variables can only be declared in the main function.

If the program file does not have a main function, then outputting a STRING using print function is NOT supported in the function scope.

NO nested function call is allowed.

There is NO return value for any function.

Function calling:
Samples of calling function are shown as follows.

```
{func1}

--or--

{func2} a

--or--

{func3} a,b
```

The function name should be contained in a pair of curly braces. Parameters are typed after the function name. If there are two, a comma should be used.

Function calls are NOT allowed inside a function.
The variables which sent as instances into the function will NOT change its value after the function finishes running. In other words, the function does not support references.

There is no return value for any function.

4. Arithmetic Expressions

Arithmetic expressions supports addition, subtraction, multiplication and division, their symbols are +, –, *, /. Brackets are also allowed. The following is an example.

```
a = (1+2) / 3 - 4 * (5+6) +7
```

The declaration of integer variable CANNOT be in the same line with arithmetic assignment.

Brackets should NOT be contained within brackets, that is, nested brackets is NOT supported.

The variable must be declared before assignment.

5. Output Function

The language supports a simple output function called print.

There are several different syntax for different purpose. Outputting an integer, a variable or a string are all supported. Samples are as follows:

```
{print} 10

--or--

{print} a

--or--

{print} 'Hello world.'
```

Strings are contained ONLY in SINGLE quotation marks, rather than double quotation marks.

Function print only supports printing one element at a time.

6. Others

Input function is not supported.

Each line of code will be considered as a complete sentence, so there is no stopping sign like ; at the end of each line.

Spaces are allowed anywhere.

Blank lines are allowed between lines, so the syntax is quite flexible.