Project Step 1 — Scanner

The first step of the project is building the first phase of a compiler: the scanner (sometimes called a tokenizer). A scanner's job is to convert a series of characters in an input file into a sequence of *tokens* -- the "words" in the program. So, for example, the input

```
A := B + 4
```

Would translate into the following tokens:

```
IDENTIFIER (Value = "A")
OPERATOR (Value = ":=")
IDENTIFIER (Value = "B")
OPERATOR (Value = "+")
INTLITERAL (Value = "4")
```

The way that we define tokens in a programming language is with *regular expressions*. For example, a regular expression that defines an integer literal token looks like:

`[0-9]+` (read: "1 or more digits"),

while a regular expression that defines a float literal token looks like:

`[0-9]+\.[0-9]* | \.[0-9]+` (read: "Either 1 or more digits followed by a decimal followed by 0 or more digits; or a decimal followed by 1 or more digits")

(See the lecture notes on Scanners for details).

While you can write a scanner by hand, it is very tedious. Instead, we typically use tools to help us *automatically generate* scanners. The tool we are using is ANTLR.

Token definitions

We will be building a compiler for a simple language called LITTLE in this class. The token definitions (written in plain English) are as follows:

```
an IDENTIFIER token will begin with a letter, and be followed by any number of letters
and numbers.

IDENTIFIERS are case sensitive.

INTLITERAL: integer number
            ex) 0, 123, 678
FLOATLITERAL: floating point number available in two different format
                yyyy.xxxxxx or .xxxxxxx
            ex) 3.141592 , .1414 , .0001 , 456.98

STRINGLITERAL: any sequence of characters except '"'
            between '"' and '"'
            ex) "Hello world!" , "***********" , "this is a string"

COMMENT:
      Starts with "--" and lasts till the end of line
      ex) -- this is a comment
      ex) -- anything after the "--" is ignored

Keywords
```

```
PROGRAM,BEGIN,END,FUNCTION,READ,WRITE,IF,ELSE,FI,FOR,ROF,RETURN,INT,VOID,STRING,FLOAT,
WHILE, ENDIF, ENDWHILE

Operators
:= + - * / = != < > ( ) ; , <= >=
```

What you need to do

You should build a scanner that will take an input file (LITTLE source program) and output a list of all the tokens in the program. For each token, you should output the token type (e.g., `OPERATOR`) and its value (e.g., `+`).

Sample inputs/outputs (i.e. testcases) are provided. Your outputs need to match our outputs *exactly* (they will be automatically compared using `diff`, though whitespace will be ignored).

Hints

While it might seem weird, you *may* need to define a token that eats up any whitespace in your program (recall that your compiler really only sees a list of characters; it has no reason to think that a tab character isn't an important character). Make sure that when you recognize a whitespace token, you just silently drop it, rather than printing it out.