COSC 341 Programming Languages Project #1

Precis: Write a program that reads a plain text source code file, inserting identifier names into a symbol table, along with the line #, the identifier's type and nesting level.

<u>More detailed description</u>: The source file program will contain identifier names, three keywords (‘`proc`’, ‘`int`’ , ‘`real`’), ‘`{`‘, ‘`}`’, ‘`;`’ , ‘=’, and constant integers.

The source file will be properly nested with ‘{‘ and ‘}’.

The nesting level starts at level 0.
When ‘{‘ is encountered, the nesting level increments by 1.
When ‘}’ is encountered, the nesting level decrements by 1.

As each identifier declaration is encountered, the identifier name, its nesting level, its type, and the line# is entered into a symbol table.

Usage of the identifier is not entered into the symbol table.
*** For extra credit, access the symbol table and add the line number(s) where the identifier is used ***

<u>Input:</u> A plain text source file (char stream). The source file will NOT contain line numbers. The line numbers are given below so that you can see what is entered to the symbol table.

The input file must be supplied as a run-time argument or as user-supplied char string to a prompt from the program. The input file may not be hard-coded in your program.

<u>Output:</u> Symbol table listing.

<u>Example: source file</u>

```
1    proc main {
2       int x;
3       real y;
4
5       proc t1 {
6         int x;
7
8         proc t2 {
9            real x;
10           int y;
11           int z;
12           x = y + z;
13          }
14          x = 3;
15        }
16        x = 5;
17     }
```

Example: Symbol table output

```
NAME TYPE  LEVEL L#
main proc 0     1
x    int  1     2
y    real 1     3
t1   proc 1     5
x    int  2     6
t2   proc 2     8
x    real 3     9
y    int  3     10
z    int  3     11
```

When a variable is declared, the type (`proc, int, real`) will precede the name.
Permitted:

```
int x
proc t1223
```

Because ',' is not permitted in the source file, each identifier will be immediately preceded by its type.

End of line character does not have semantic significance. So

```
     int x; real counter; int tester;
```

is valid

Language of implementation: Your choice. If you use perl, your code **must** be easily readable.

Grammar for source file will follow.

Inputs: You must put each of these inputs into a separate file.

Input #1

```
proc g {
   int x;
   int y;
   {
      real x;
      real z;
   }
}
```

<u>Input # 2</u>

```
proc f1 {
   int x;
   {
      int y;
      {
         int z;
         x = z + 25;
      }
   }

   proc f2 {
      real y;
      x = y;
   }

   real z;
   z = 3;
   x = z;
}
```

<u>Turn in:</u>
Hard copy of source code
Hardcopy of Example file, Input #1 file, Input #2 file.
Screen shot of program working on Example, Input #1, Input #2
Code walk-through.

<u>Extra-credit</u>

For Example code, the symbol table output should be:

```
NAME TYPE  LEVEL L#
main proc  0     1#
x    int   1     2# 16
y    real  1     3#
t1   proc  1     5#
```

```
x    int  2    6# 14
t2   proc 2    8#
x    real 3    9# 12
y    int  3    10# 12
z    int  3    11# 12
```

Note, the line where the variable is declared is followed with ‘#’. Lines where variables are used do not have any special symbol.