CS4430 HOMEWORK 1 (100 POINTS TOTAL)

Issued: Monday, February 13th, 2017.
Due: Wednesday, February 22nd, 2017 ~~Monday, February 20th, 2017~~ by 11:59pm.

DIRECTIONS

To turn in your solution, please email me the code (harrisonwl@missouri.edu) with the subject "CS4430 HW1". It is **important** that you get this small detail right because otherwise I may miss your submitted solution.

This programming assignment has two purposes: firstly, to help you dust off your knowledge of Haskell, and, secondly, to expose you to the issues involved in lexical analysis.

PROBLEM DESCRIPTION

Below is an informal description of the syntax of a compiler intermediate representation called Tuple. Each instruction has the form: (OPCODE, arg_1,…,arg_n), where OPCODE is the particular instruction and arg_1,…,arg_n is an operand sequence (possibly empty) separated by commas. Operands are registers, integer literals, or memory references.

OPCODES are: **`ASSIGN WRITEI READI ADDI SUBI JUMP JNZ LABEL`**

REGISTERS are: **`SP BP FP R`***n* where *n* is any non-negative integer with no leading **0**'s

LITERALS are: any nonempty string of digits **0** through **9** with no leading **0**'s

MEMREFS are: **`M[`***arg***`]`** where *arg* is either a register, a register **+** a literal, or a literal.

Below is some sample Tuple code.

```
(ASSIGN,BP,0)
(ASSIGN,SP,0)
(ASSIGN,FP,SP)
(ASSIGN,M[FP],3)
(ADDI,SP,FP,2)
(JUMP,2)
(LABEL,0)
(ADDI,SP,SP,1)
(ADDI,SP,SP,1)
(ASSIGN,R0,M[FP+4])
(READI,M[FP+4])
(ASSIGN,R1,M[FP+5])
(ASSIGN,R2,4)
(ASSIGN,M[FP+5],R2)
(ASSIGN,M[SP],4)
(ADDI,SP,SP,1)
(ASSIGN,M[SP],FP)
(ADDI,SP,SP,1)
(SUBI,M[SP],SP,2)
(ADDI,SP,SP,1)
(ASSIGN,R3,M[FP+4])
(ASSIGN,R4,R3)
(ASSIGN,M[SP],R4)
(ADDI,SP,SP,1)
(ASSIGN,R5,M[FP+5])
(ASSIGN,R6,R5)
(ASSIGN,M[SP],R6)
(ADDI,SP,SP,1)
(SUBI,FP,SP,5)
(JUMP,1)
(LABEL,4)
(ASSIGN,SP,M[FP+2])
(ASSIGN,FP,M[FP+1])
(JUMP,M[FP])
(LABEL,1)
(ASSIGN,R7,M[FP+3])
(ASSIGN,R8,R7)
(ASSIGN,R9,M[FP+4])
(ASSIGN,R10,R9)
(ADDI,R11,R8,R10)
(WRITEI,R11)
(JUMP,M[FP])
(LABEL,2)
(ADDI,SP,SP,1)
(ASSIGN,R12,M[FP+5])
(READI,M[FP+5])
(ASSIGN,R13,M[FP+5])
(ASSIGN,R14,R13)
(WRITEI,R14)
(ASSIGN,R15,M[FP+6])
(ASSIGN,R16,R15)
(WRITEI,R16)
(ASSIGN,M[SP],5)
(ADDI,SP,SP,1)
(ASSIGN,M[SP],FP)
(ADDI,SP,SP,1)
(SUBI,M[SP],SP,2)
(ADDI,SP,SP,1)
(ASSIGN,R17,M[FP+5])
(ASSIGN,R18,R17)
(ASSIGN,M[SP],R18)
(ADDI,SP,SP,1)
(SUBI,FP,SP,4)
(JUMP,0)
(LABEL,5)
(ASSIGN,SP,M[FP+2])
(ASSIGN,FP,M[FP+1])
(JUMP,M[FP])
(LABEL,3)
```

PROBLEM ASSIGNMENT

Your job is to write a simple lexer for this language using the pattern discussed in class that uses the Parsing.lhs library. Your solution will look quite similar to MicroLex.hs, so I'd recommend studying that carefully.

Write a function, `tuplelex :: String -> Maybe [Token]`, so that if `test` is defined as below, `tuplelex` produces the following output:

```
ghci> test
   "(ASSIGN,BP,R99)\n (ASSIGN,SP,0)\n (JUMP,M[FP])\n (LABEL,3) "
ghci> tuplelex test
   Just [LPAREN,ASSIGN,COMMA,BP,COMMA,R "99", RPAREN, LPAREN, ASSIGN,
         COMMA, SP, COMMA, LITERAL"0", RPAREN, LPAREN, JUMP, COMMA, M,
         LBRACKET, FP, RBRACKET, RPAREN, LPAREN, LABEL, COMMA,
         LITERAL "3", RPAREN, SCANEOF]
```

Finally, when the end of the input is reached, notice that the `SCANEOF` token is emitted.

A few nerdly requirements here:

1. Whitespace (i.e., blanks and carriage returns) doesn't matter. For example, "`( ASSIGN ,R99 ,M [FP ] )`" must give the same output as "`(ASSIGN,R99,M[FP])`".
2. All legal `OPCODE`s and REGISTERs are listed above.
3. Case matters!
4. Errors can occur. For example, any register not in the appropriate form (e.g., "EAX") listed above must cause your `tuplelex` function to return `Nothing`.
5. This program must be written in Haskell. You should use the most recent version of the Haskell Platform. Make sure that your solution works with the current version of the Haskell platform.
6. You should put all the code you write in a file entitled "`CS4430_HW1_`*yourlastname*`.hs`".
7. Remember to test your solution well – I will!
8. Don't wait until the last minute!