1X1: PROGRAMMING LANGUAGES

Due 8/14 in the Assignment 1 Dropbox on E3 at 11:45PM

OVERVIEW

As mentioned in lecture, all of the languages we will study this quarter are interpreted.

An interpreter translates programs written in one language into computational actions, effectively running the program being interpreted. Code Analysis is the first phase in Interpretation, which ensures the source program is syntactically and semantically correct.

This assignment requires you to implement the syntactic analysis phase of Interpretation without the need for error checking.The interpreter's computational engine will be implemented in a future assignment.

SIMPLESEM INSTRUCTION FORMAT

The interpreter is being written for a language named SIMPLESEM. The SIMPLESEM instruction set contains four op-codes: `set, jump, jumpt, and halt`. The instruction formats are as follows (a future assignment specification will explain each instruction's semantics):

```
set destination, source
jump destination
jumpt destination, boolean-condition
halt
```

To perform syntactic analysis of SIMPLESEM you will:

- Open & Read a SIMPLESEM program
- Use Python to identify & tokenize keywords, symbols, and expressions
- Use the SIMPLESEM grammar below to parse a program according to language rules

SIMPLESEM GRAMMAR

The grammar for SIMPLESEM is as follows:

<Program> → <Statement> {<Statement>}
<Statement> → <Set> | <Jump> | <Jumpt> | **halt**
<Set> → **set** (**write**|<Expr>)**,** (**read**|<Expr>)
<Jump> → **jump** <Expr>
<Jumpt> → **jumpt** <Expr>**,** <Expr> (**!=** | **==** | **>** | **<** | **>=** | **<=**) <Expr>
<Expr> → <Term> {(**+** | **-**) <Term>}
<Term> → <Factor> {(***** | **/** | **%**) <Factor>}
<Factor> → <Number> | **D[**<Expr>**]** | **(**<Expr> **)**
<Number> → **0** | (**1**..**9**){**0**..**9**}

Guide to EBNF: |— separate alternate choices, ()—choose 1, {}—choose 0 or more, keyword/symbol—what is in **bold blue Courier New** font.

SYNTACTIC ANALYSIS

To parse the file according to the SIMPLESEM grammar, you will need to complete the following tasks :

- Open input file for reading (NAME: provided as command-line argument; FORMAT: one SIMPLESEM statement per line)
- As described in lecture & lab, use recursion and looping to correctly parse SIMPLESEM grammar rules.

Note: Since the testcases are syntactically correct, error checking code is not needed.

OUTPUT

Print the name of each non-terminal entered. Please see the sample .out files for example output based on each test case.

SUBMISSION

Submit the following items to the Assignment #1 Dropbox on E3:

- A single zip file named 1X1_Lab1 which should contain ONLY files needed to implement this project.

All projects must be able to execute via command line receiving the filename as a command line argument:

FOR PYTHON: python SIMPLESEM.py Program#.S
FOR C++: g++ SIMPLESEM.cpp -o SIMPLESEM
SIMPLESEM.exe Program#.S

NOTE: Under no circumstances should you upload an executable.