CSE 465/565
Fall 2018
Homework #1
100 points

Instructions: Submit to Canvas a single zip file that contains an electronic copy of your answers and programs. The zip file must have the following directory structure, where uniqueid is your Miami uniqueid:

uniqueidHW1 *top-level directory containing all of your stuff*
uniqueidHW1.pdf *; which contains the output for exercise (1) and report for exercise (2)*
uniqueidL *directory containing your source code for exercise (2)*
main.cpp, helper.cpp, etc *; your source code in either C++ or Java*
Main.java, Helper.java, etc *; please, do not put your code into packages*

For instance, if your uniqueid were inclezd *and your solution for exercise (2) were written in Java, your zip file should contain the following directory structure:*

inclezdHW1
inclezdHW1.pdf
inclezdL
Main.java, Helper.java, etc.

Make sure that your main file is called either Main.java or main.cpp, depending on the language you used for the implementation. No other file names are allowed for the main file.

For each exercise, you will see the points in parenthesis. The first number is the number of points for CSE 465 students; the second number is the number of points for CSE 565 students.

(1). (10/10) Log into `ceclnx01.csi.miamioh.edu` and create a new folder for this class. Copy the file `test.txt` provided to you into this new folder. On the `ceclnx01` server, run the following three commands:

```
dlv test.txt
time dlv test.txt
wc test.txt
```

Type the commands exactly as you see them above. Take a snapshot of the commands' output and insert it into your PDF.

(2). (90/90) Consider a simple programming language named `L`. The `L` programming language has the following features:

- `L` variables can store a string, a Boolean, or an integer value. A single variable can switch between string, Boolean, and integer values during program execution. Assigning a value to a variable creates that variable for future use. A runtime error occurs if a variable is used before it is given a value. Boolean values are denoted by the reserved words TRUE and FALSE.

- Variables are **case sensitive** and consist only of upper and lower-case letters.

- The following are L reserved words: PRINT, FOR, TRUE, FALSE, WHILE, DO, LESSTHAN plus the reserved symbols { and }. LESSTHAN, WHILE, and DO are only relevant for graduate students.

- You may **assume** that any L program given as an input to your interpreter is **syntactically correct**.

- The PRINT statement should display one variable's value. Here is an example of a PRINT statement in language L:

```
PRINT cookies ;
```

  If cookies is an integer variable with value 4, the output of your interpreter should be:

```
cookies=4
```

  If cookies is a Boolean variable with value TRUE, the output of your interpreter should be:

```
cookies=TRUE
```

  If cookies is a string variable with value "chocolate chip", the output should be:

```
cookies="chocolate chip"
```

- The left-hand side of an assignment statement (i.e., =) is a variable.

- The right-hand side of a **simple** assignment statement (i.e., =) is either a variable name, a signed integer, string literal, or the Boolean values TRUE or FALSE. If the right-hand side of the assignment is a variable, then this variable must already have a value; otherwise the assignment should trigger a runtime error to be reported by your interpreter. For example, the following are valid assignments:

```
B = 0 ;
A = 12 ;
A = B ;
A = "hello" ;
A = "12" ;
A = TRUE ;
```

  The following is not a valid assignment, assuming C has no value yet:

```
A = C ;
```

- There are three **compound** assignment statements: +=, *=, and &=. The meaning of these operators depends on the data type of the left and right-hand side of the operator.

```
<string var> += <string>      concat right string onto end of left string
<integer var> += <integer>    increment left integer with value on right
<integer var> *= <integer>    multiply left integer by value on right
<Boolean var> &= <Boolean>    assign to left Boolean the result of performing
                              a Boolean AND operation between values on left
                              and right
```

Examples:

```
A = 4 ;
A += 34 ;
A *= B ;

C = "t" ;
C += "hello world" ;

D = FALSE ;
D &= TRUE ;
```

All other combinations (e.g., += with string var on the left and integer on the right) are illegal and should cause a runtime error.

- Every statement is terminated by a semi-colon.

- L programs must have at least one space separating all lexical elements.

- There is a for loop statement – FOR – whose body contains at least one simple statement (i.e., at least one assignment), which is presented on one line. The keyword FOR is followed by an integer constant, which indicates the number of times to execute the loop. Following this number is the reserved symbol { followed by a sequence of statements defining the loop's body and ending with the reserved symbol }, as done here:

```
FOR 5 { B += A ; A *= 2 ; }
```

- L for loops can be nested and must appear on one line:

```
FOR 5 { B += A ; A *= 2 ; FOR 10 { A += B ; } A += 2 ; }
```

- Here is an example L program:

```
A = 1 ;
B = 0 ;
FOR 5 { B += A ; A *= 2 ; }
A += 1000 ;
PRINT A ;
PRINT B ;
```

This program's output is:

```
A=1032
B=31
```

As an example, an equivalent Python program would be (Do not translate L code into Python!):

```
A = 1
B = 0
for i in range(5):
```

```
   B += A
   A *= 2
A += 1000
print("A=" + str(A))
print("B=" + str(B))
```

Here is a second `L` program:

```
A = 10 ;
A += A ;
PRINT A ;
A = "hello" ;
A += A ;
PRINT A ;
A += 123 ;
PRINT A
```

The output to this second program would yield an error. Your program should display the line number of this error and then stop processing:

```
A=20
A="hellohello"
RUNTIME ERROR: line 7
```

- **Requirements for graduate students only:** Graduate students must additionally support:
  - An assignment statement of the form shown below, where the left-hand side is a Boolean variable:

    ```
    <Boolean var> = <integer> LESSTHAN <integer>
    ```

    Examples:

    ```
    A = 5 ;
    B = 4 ;
    C = A LESSTHAN B ;
    C = A LESSTHAN 0 ;
    ```

  - While loops, which are statements consisting of the reserved word `WHILE` followed by a Boolean variable, followed by the reserved word `DO`, and ending with the loop body. The loop body starts with the reserved symbol `{`, continues with a sequence of statements, end ends with the reserved symbol `}`. A whole while loop statement must appear on the same line in an `L` program.

    Example:

    ```
    A = 3 ;
    B = TRUE ;
    WHILE B DO { A += 5 ; B = A LESSTHAN 15 ; }
    ```

Notes:

- You may assume that the programs are syntactically correct but may have runtime errors (e.g., add integer and string, use a variable without a value on the right-hand side of an assignment, etc.).
- Your program must run on `ceclnx01` using the standard compile command. The name of the `L` program file will be passed to your program using a **command line argument**:

```
c++ -std=c++11 -o myapp *.cpp
myapp prog1.txt

javac *.java
java Main prog1.txt
```

Tasks and scoring:

- (80/80 points) Write an interpreter (Java or C++) to execute `L` programs
  - (30/10 points) Basic structure, integer variables only
  - (25/25 points) Basic structure, integer, plus string and Boolean variables
  - (10/5 points) For loops
  - (10/20 points) Nested for loops
  - (5/5 points) Detection of runtime errors
  - (0/5) Boolean assignments with LESSTHAN comparison
  - (0/10) While loops
- (10/10 points) Write a report that:
  - Explicitly states what works in your interpreter and what does not.
  - Provides the output of your interpreter when run on the sample `L` programs provided in the "Sample Programs.zip" archive in Canvas.
  - Compares the runtime speed of `L` programs versus an equivalent program in some **compiled** language (e.g., C++). Provide timings to support your estimate. Are your findings typical of interpreted languages? Describe how you set up the experiment in order to obtain meaningful timings (e.g., what `L` programs you used for testing – note that you can write your own `L` programs).