CSE 465/565
Spring 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 your Z+- report and the output for exercise (2) below*
uniqueidZPM *directory containing your source code*
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 (3) were written in Java, your zip file should contain the following directory structure:*

inclezdHW1
inclezdHW1.pdf
inclezdZPM
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.

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). (5/5) Log into https://miamioh.qualtrics.com/jfe/form/SV_ah4Ilk3HySh5D7L and take the pre-semester outcome survey. **This must be done by Friday, February 2nd 2018, at 5:00pm.** The survey is anonymous, in that I am unable to view your answers. However, I will know if you have completed the survey.

(2). (10/10) Log into `ceclnx01.csi.miamioh.edu` and create the following Java file - `Sample.java` .

```
import java.util.*;
public class Sample {
    public static void main(String [] args) {
        Scanner input = new Scanner(System.in);
        String user = System.getProperty("user.name");
        System.out.println(user);
        int printed = 0;
        while (input.hasNext() && printed < 10) {
            String line = input.nextLine();
            if (line.indexOf("inclezd") == 0 ||
                line.indexOf(user) == 0) {
                    System.out.println(line);
                    printed++;
            }
        }
        input.close();
    }
}
```

Compile and run the program with the following commands:

```
javac Sample.java
last | java Sample
```

Copy the commands exactly as you see them above: `last` is a command that shows information about the last logged in users. Insert the program's output into your PDF.

(3). (85/85) Consider a very simple programming language named `Z+-`. The `Z+-` programming language has the following features:

- Z+- variables can store a string or integer value. A single variable can switch between integer and string 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.
- Variables are **case sensitive** and consist only of upper and lower-case letters.
- The following are Z+- reserved words: `PRINT, FOR, ENDFOR, PROC, ENDPROC, CALL`
- You may **assume** that the Z+- program is **syntactically correct**.
- The PRINT statement displays one particular variable's value. This is done as:

```
PRINT numCookies ;
```

- The right-hand side of a simple assignment statement (i.e., =) is either a variable name (which must have a value), signed integer, or string literal. For example, the following are valid:

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

- 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
<integer var> -= <integer> subtract right integer from value on left
```

```
A += 34 ;
A *= B ;
A += "hello world" ;
```

All other combinations are illegal and cause a runtime error.

- Every statement is terminated by a semi-colon.
- There is a loop statement – `FOR` – whose body contains at least one simple statement (i.e., at least one assignment), which are 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 a sequence of statements defining the loop's body, followed by the word `ENDFOR`, as done here:

```
FOR 5 B += A ; A *= 2 ; ENDFOR
```

- `Z+-` for loops can be nested and must appear on one line:

```
FOR 5 B += A ; A *= 2 ; FOR 10 A += B ; ENDFOR A += 2 ; ENDFOR
```

- Z+- programs must have at least one space separating all lexical elements.
- Here is an example Z+- program:

```
A = 1 ;
B = 0 ;
FOR 5 B += A ; A *= 2 ; ENDFOR
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 Z+- 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 Z+- 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
```

- **Requirement for graduate students only:** Graduate students must support parameterless procedures. Any procedures must be defined prior to the "main program statements". The procedures must have unique names and contain a list of legal Z+- statements. Recursion is not allowed. All variables created/used are those at the global level. Here is an example:

```
PROC F
A *= 2 ;
B *= 2 ;
ENDPROC

PROC G
C += 1 ;
CALL F ;
ENDPROC

A = 1 ;
B = 1 ;
C = 0 ;
CALL F ;
FOR 10 CALL G ; ENDFOR
```

Notes.

- You may assume that the programs are syntactically correct but may have runtime errors (e.g., add integer and string).
- Your program must run on ceclnx01 using the standard compile command. The name of the ZPM file will be passed to your program using a command line argument:

```
c++ -std=c++11 -o myapp *.cpp
myapp prog.zpm

javac *.java
java Main prog.zpm
```

Tasks and scoring:

- (75/75 points) Write an interpreter (Java or C++) to execute `Z+-` programs
  - (30/10 points) Basic structure, integer variables only
  - (20/20 points) Basic structure, integer AND string variables
  - (10/5 points) For loops
  - (10/20 points) Nested for loops
  - (5/5 points) Detection of runtime errors
  - (0/15) Parameterless procedures
- (10/10 points) Write a report that:
  - Explicitly states what works in your program and what does not.
  - Provides the output of your program when run on the sample `Z+-` programs provided in the "Sample Z+-programs.zip" archive in Canvas. Identify any incorrect output that your program produces.
  - Compares the runtime speed of `Z+-` 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?