Benjamin Bowser
CSE 465
HW1
February 12th, 2018

2.

```
bowserbl@ceclnx01:~/465$ last | java Sample
bowserbl pts/3        173.244.44.40    Thu Feb  8 18:18   still logged in
inclezd  pts/27       10.33.3.116      Wed Feb  7 16:04 - 17:44  (01:40)
inclezd  pts/29       10.33.3.116      Wed Feb  7 14:46 - 14:47  (00:01)
inclezd  pts/2        10.33.3.116      Fri Feb  2 17:48 - 17:48  (00:00)
inclezd  pts/3        10.33.3.116      Fri Feb  2 16:50 - 16:52  (00:02)
inclezd  pts/26       10.33.3.116      Fri Feb  2 15:10 - 16:49  (01:39)
inclezd  pts/1        10.32.1.99       Fri Feb  2 14:11 - 14:14  (00:02)
```

3.

In this program, I found that the structures and strings worked without any problems at all. After working on the program more, I was able to integrate the ability to use integers along with strings. I stored everything in a hashmap with the key and values both being strings. For integer mathematics, I extracted the numbers from the string and converted them to integers, did any necessary computations, then wrote them back as a string. This method allowed strings to work flawlessly with the program. I found for loops to be working as expected, in addition, I observed that some of the given programs had errors in the for loops, which it caught and displayed a runtime error. Nested for loops work in some instances, but not all. I found that it depended on the location of the ENDFOR in the statement in regard to the overall success or not. I managed to code runtime errors to catch exceptions as they occur. I am not a graduate student, so I did not implement parameterless procedures.

Output:

Prog1.zpm:

```
a=4
b=360
```

Prog2.zpm:

```
RUNTIME ERROR: line 10
```

Prog3.zpm:

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

Prog4.zpm:

```
A=XXXXXXXXXXXXXXX
```

Prog5.zpm:

```
RUNTIME ERROR: line 7
```

Prog6.zpm:

```
A=107
B=10
C=3
D=4
E=214
```

Prog7.zpm:

```
RUNTIME ERROR: line 3
```

Prog8.zpm:

```
RUNTIME ERROR: line 2
```

Prog9.zpm:

```
A=0
a=5
RUNTIME ERROR: line 5
```

Prog10.zpm:

```
A=1
a=3
numItems=0
```

Time comparison:

I wrote my interpreter in Java, so I figured that it would be good to write an equivalent program in Java to test speeds. I took program 1, which is a basic program that would be taught in an introductory level programming course and converted it into Java. I added a timer to my z+- program to test the speeds of the program. The program took 0.021 seconds to run on my computer. I also ran my same code that was written in Java, wrapped inside of a timer. I observed that on every run, the program outputted 0.0 seconds for runtime, not even enough time to register on the timer. These findings support the idea that interpreted languages can be slower. Z+- code has to be run through my interpreter, which is being run by Java. For the Z+- code to run, both Java and the interpreter need to run. For simple execution of my code in Java that was used to compare, it skips the interpretation step and just uses Java. By only using Java, the code runs much faster. I would expect the exact same findings if this was being tested on a language like C++.