LAB #10
Wrapping Up the Fall Quarter

Check your grades on Canvas to make sure they are accurate!!! We do not have more labs, and you need to make sure your grade is correct!!!

Make sure you have Assignments #1 - #9 graded because this is the last week the TAs will hold office hours for grading or helping anyone!!! We will grade Assignment #10 during finals week. You do not need to demo Assignment #10!!!

You can pair with someone or do this on your own.

These are two example problems are taken from a list of 46 simple python exercises: http://www.ling.gu.se/~lager/python_exercises.html
If you finish early try more of these on your own. 😊

(5 pts) Code the design/pseudocode

```
# Description: This is where execution of the program begins. The main function will get
# the number of verses to sing as a command line argument, and then sing the song
# with that many verses.
# Parameters: none
# Preconditions: none
# Postconditions: none
# Return: none
main()
    Make sure there is a command-line argument
    Read the number of verses as a string
    if the string is an integer, i.e. send string to is_positive_int()
        send the converted integer to sing_song()
    else
        print an error message

# Description: This function checks to see if a string is an int
# Parameters: s
# Preconditions: s is a string
# Postconditions: none
# Return: bool, true if it is an int and false otherwise
is_positive_int(s)
    for all the characters in s
        if (the character is not between ‘0’ and ‘9’ inclusive)
            return false
    return true
```
# Description: This function sings the song, "99 bottles of beer on the wall", but it starts
# at a specific verse/number.
# Parameters: n
# Preconditions: n is an integer value
# Postconditions: none
# Return: none

```python
sing_song(n)
    while n verses is greater than 0
        print "n bottles of beer on the wall. Take one down, pass it around. n-1 bottles of beer on the wall."
        decrement n by 1
```

(5 pts) Code the design/flowchart

Description - this is where execution begins and it gets a string from the user and checks if the string is balanced parenthesis parameters, preconditions, postconditions, & return.

```
main()
    get string of parenthesis from file
    if string is balanced yes is_balanced(s)
        print yes message
        no
        print no message
```
Description: checks to see if string is balanced parenthesis. For example: 

( ), ((())), ()() are balanced 

(, ), ), )(, )(()()), )()(' are not balanced

Parameters: s
Pre conditions: none
Post conditions: none
Return: True if string is balanced parenthesis, False otherwise

`is_balanced(s)`

1. Set open-paron to 0
2. If we are at the end of the string:
   - No
     - If the character is ')' and open-paron is zero:
       - Yes
         - Return False
       - No
         - If the character is '(': Increment open-paron by one
         - Yes
           - Return False
         - Else
           - Else if character is ')': Decrement open-paron by one
3. Yes
   - If open-paron to 0:
     - Yes
       - Return True
   - No
     - Return False