LAB #10  
Wrapping Up the Fall Quarter

Check your grades on BB 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](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 from the user, and then sing the song with that many
# verses.
# Parameters: none
# Preconditions: none
# Postconditions: none
# Return: none
main()
    Prompt the user for the number of verses
    Read the number of verses as a string
    if the string is an integer, send string to is_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_int(s)
    for all the characters in s
       if ((the character is not between '0' and '9' inclusive) and (it is not the first))
          or (it is the first character and not equal to '-'))
          return false
    return true
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

```
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
    if string is balanced is_is_balanced(s) yes
    print yes message
    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)

set open-paron to 0

if we are at the end of the string

yes

if open-paron to 0

yes

return False

no

return True

no

if the character is ')' and open-paron is zero

yes

return False

no

decrement open-paron by one

yes

else if the character is ')

no

increment open-paron by one

yes

else if character is '('

no

return False