NAME:
EMAIL:
SIGNATURE:
CIRCLE COURSE SECTION: MW 9-11 MW 11-1 MW 6-8
TTH 9-11 TTH 1-3

Lehman College, CUNY
CIS 166 Final Exam, Version 3, Spring 2015

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
s = "haskell::curry::utrecht::glasgow"
a = s[0:3]
print(a.upper())
names = s.split("::")
print(names)
b,c,d = names[1],names[2],names[3]
print(c,d)
print(a[0]+b.upper(), "versions:")
print(c.title(),d.capitalize())
print('main = putStrLn "', names[0],'"')
```

Output:

2. Write a **complete program** to calculate how much something will weigh on Jupiter. Your program should prompt the user for the weight on the Earth and then print out the weight on Jupiter. For example, if the user enters `100`, your program should print out `254`.

   The weight of an item on Jupiter is 254% *of its weight on earth.*

3. What is output of the code below:

```
def prob4(paul, john):
     if paul > 2:
          print("Easy case")
          yoko = -1
     else:
          print("Complex case")
          yoko = helper(paul,john)
     return(yoko)
```

```
def helper(ringo,george):
     s = ""
     for j in range(ringo):
          print(j, ": ", george[j])
          if j % 2 == 0:
               s = s + george[j]
               print("Building s:", s)
     return(s)
```

(a)
```
r = prob4(6,"city")
print("Return:  ", r)
```

Output:

(b)
```
r = prob4(2,"university")
print("Return:  ", r)
```

Output:

(c)
```
r = prob4(4,"new york")
print("Return:  ", r)
```

Output:

4. Given the following program and input file, what is printed:

```
def prob5V1():
   c = 0
   infile=open("places.txt","r")
   for line in infile.readlines():
      if len(line)>9:
         print("Long Line: ", end ="")
         c = c+1
      print(line)
   print("Num long lines is", c)

prob5V1()
```

places.txt

```
Santo Domingo
Santiago
San Cristobal
Puerta Plata
La Vega
La Altagracia
```

Output:

5. (a) Write a function that takes number between 1 and 7 as a parameter and returns the corresponding number as a string. For example, if the parameter is 1, your function should return `"one"`. If the parameter is 2, your function should `"two"`, etc. If the parameter is not between 1 and 7, your function should return the empty string.

(b) Write a `main()` that allows the user to enter a number and calls your function to show that it works.

6. Complete the following program, which sets up a graphics window and turtle, draws two squares to the window, and then prints a closing message and closes the graphics window when mouse is clicked. That is, write the functions `setUp()`, `draw2Squares()`, and `conclusion()`:

```
import turtle

def main():
    w,t = setUp()   #sets up a graphics window and turtle
    draw2Squares(t)  #draws 2 squares using the turtle
    conclusion(w)   #prints goodbye and closes window on click

main()
```

7. (a) Write a **complete** program that prompts the user for a file name and prints the number of lines in the file.

(b) Write a **complete** program that prints the population stored in a data file. Your program should open the file, `cityData.csv` and sum the last values in each line. Note that the first line should not be used since it contains the column headers and not data. The data is separated by commas ("**,**"). Your program should print the running sum that you calculated.

cityData.csv:

```
Borough, County, Population
Bronx, Bronx, 1385108
Brooklyn, Kings, 2504700
Manhattan, New York, 1585873
Queens, Queens, 2230722
Staten Island, Richmond, 468730
```

8. Write the Python code for the algorithms below:

(a)
```
getInput()
Ask user for a positive number
Until they enter a positive number
    Print error message
    Ask user for a positive number
Return the positive number entered
```

(b)
```
sort(ls)
Set L to be the length of the list ls.
For i = 0,1,...,L-2:
    For j = 0,1,...,L-2:
        If ls[j] is bigger than ls[j+1], swap the values
Return the list, ls.
```

9. In lab, we wrote a Tic-Tac-Toe program. Modify the program to stop the game when someone has won. Your program should check for a winner each move. Your program should continue playing until there is a winner or until all squares are filled.

   Clearly mark your changes to the design below:

```
#Second Version of Tic-Tac-Toe
from turtle import *
def setUp():
    win, tic = Screen(), Turtle()
    tic.speed(10)
    win.setworldcoordinates(-0.5,-0.5,3.5, 3.5)
    for i in range(1,3):
        tic.up()
        tic.goto(0,i)
        tic.down()
        tic.forward(3)
    tic.left(90)
    for i in range(1,3):
        tic.up()
        tic.goto(i,0)
        tic.down()
        tic.forward(3)
    tic.up()
    board = [["","",""],["","",""],["","",""]]
    return(win,tic,board)
def playGame(tic,board):
    for i in range(4):
        x,y = eval(input("Enter x, y coordinates for X's move: "))
        tic.goto(x+.25,y+.25)
        tic.write("X",font=('Arial', 90, 'normal'))
        board[x][y] = "X"
        x,y = eval(input("Enter x, y coordinates for O's move: "))
        tic.goto(x+.25,y+.25)
        tic.write("O",font=('Arial', 90, 'normal'))
        board[x][y] = "O"
    x,y = eval(input("Enter x, y coordinates for X's move: "))
    tic.goto(x+.25,y+.25)
    tic.write("X",font=('Arial', 90, 'normal'))
    board[x][y] = "X"
def checkWinner(board):
    for x in range(3):
        if board[x][0] != "" and (board[x][0] == board[x][1] == board[x][2]):
            return(board[x][0])  #we have a non-empty row that's identical
    for y in range(3):
        if board[0][y] != "" and (board[0][y] == board[1][y] == board[2][y]):
            return(board[0][y])  #we have a non-empty column that's identical
    if board[0][0] != "" and (board[0][0] == board[1][1] == board[2][2]):
        return(board[0][0])
    if board[2][0] != "" and (board[2][0] == board[1][1] == board[2][0]):
        return(board[2][0])
    return("No winner")
def main():
    win,tic,board = setUp()   #Set up the window and game board
    playGame(tic,board)       #Ask the user for the moves and display
    print("\nThe winner is", checkWinner(board))  #Check for winner
```

10. (a) In lab, we processed name data maintained by the Social Security Administration. Write a **function** that takes as input a string of Social Security Administration name data and returns the first letter of the name. For example, your function would return the number `'H'` when given the first line of the sample file.

    Here are some sample lines from the `NY.txt` file containing the data for New York State:

    NY.txt

    ```
    NYM1910Herbert,83
    NYM1910Leo,80
    NYM1910Andrew,79
    NYM1910Ernest,79
    NYM1910Milton,79
    ```

    (b) Write a **complete program** that uses your function above to count the overall number of entries for names beginning with `'A'`. Your program should open the file `NY.txt`, keep a running total of the number of times the names start with `'A'`, and print the results.

Useful String Methods: (from p 140 of textbook)

Function	Meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` is centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of `sub` in `s`.
`s.find(sub)`	Find the first position where `sub` occurs in `s`.
`s.join(list)`	Concatenate `list` into a string using `s` as a separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` with all characters converted to lowercase.
`s.lstrip()`	Copy of `s` with leading whitespace removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns rightmost position.
`s.rjust(sub)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing whitespace removed.
`s.split()`	Split `s` into a list of substrings.
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Useful Turtle Methods: (from `http://docs.python.org/3.0/library/turtle.html`)

Function	Meaning
`t.forward(d)`	Move turtle forward `d` steps
`t.backward(d)`	Move turtle backward `d` steps
`t.right(angle)`	Turn turtle `angle` degrees to the right
`t.left(angle)`	Turn turtle `angle` degrees to the left
`t.up()`	Pull the pen up: no drawing when moving
`t.down()`	Pull the pen down: drawing when moving
`t.color(c)`	Change pen color to color c
`t.goto(x,y)`	Move turtle to coordinates `(x,y)`
`w.bgcolor(c)`	Change background color to color c
`w.setworldcoordinates(x1,y1,x2,y2)`	Resize drawing area with lower left corner as `(x1,y1)` and upper right corner `(x2,y2)`
`w.exitonclick()`	Closes graphics window on mouse click