Answer Key: CIS 166 Final Exam, Version 2, Spring 2015

1. What will the following code print:

```
s = "oBJcBJaBJmBJl"
a = s[0:3]
print(a.title())
names = s.split("BJ")
print(names)
b,c,d = names[1],names[2],names[3]
print(c,d)
print(a[0]+b.upper()+c+d+names[4])
print(’print_endline "’, a.lower(),’"’)
```

Answer Key:

```
Obj
[’o’, ’c’, ’a’, ’m’, ’l’]
a m
oCaml
print_endline " obj "
```

2. Write a **complete program** to calculate how much something will weigh on the Moon. Your program should prompt the user for the weight on the Earth and then print out the weight on the Moon. For example, if the user enters 100, your program should print out 17.

The weight of an item on the Moon is 17% *of its weight on earth.*

Answer Key:

```
#Computes weights on the moon
def main():
     earthWeight = eval(input(’Enter earth weight: ’))
     moonWeight = earthWeight * 0.17
     print(’The weight on the Moon is:’, moonWeight)

main()
```

3. What is output of the code below:

```
def prob4(amy, beth):
     if amy > 4:
          print("Easy case")
          kate = -1
     else:
          print("Complex case")
          kate = helper(amy,beth)
     return(kate)
```

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

(a)

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

Output:

Answer Key:

```
Easy case
Return:  -1
```

(b)

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

Output:

Answer Key:

```
Complex case
0 :  u
Building s: u
1 :  n
Return:  u
```

(c)

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

Output:

Answer Key:

```
Complex case
0 :  n
Building s: n
1 :  e
2 :  w
Building s: nw
3 :
Return:  nw
```

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)<10:
         print("Short Line: ", end ="")
         c = c+1
      print(line)
   print("Num short lines is", c)

prob5V1()
```

places.txt

```
Ontario
Quebec
Nunavut
Yukon
Alberta
New Brunswick
```

Output:

Answer Key:

```
Short Line: Ontario

Short Line: Quebec

Short Line: Nunavut

Short Line: Yukon

Short Line: Alberta

New Brunswick

Num short lines is 5
```

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

   Answer Key:

```
def returnNumString(num):
    if num == 1:
        return "first"
    elif num == 2:
        return "second"
    elif num == 3:
        return "third"
    elif num == 4:
        return "fourth"
    elif num == 5:
        return "fifth"
    elif num == 6:
        return "sixth"
    elif num == 7:
        return "seventh"
    else:
        return -1
```

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

   Answer Key:

```
#intro comment
def main():
    num = eval(input("Enter a number"))
    test1 = returnNumString(num)
```

```
        print ("Testing my function:",num,"is", test1)
    main()
```

6. Complete the following program, which sets up a graphics window and turtle, draws an octagon (8-sided figure) to the window, and then prints a closing message and closes the graphics window when mouse is clicked. That is, write the functions `setUp()`, `drawOctagon()`, and `conclusion()`:

```
import turtle

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

main()
```

Answer Key:

```
def setUp():
    trey = turtle.Turtle()
    win = turtle.Screen()
    return(win,trey)

def drawOctagon(t):
    for i in range(8):
        t.forward(100)
        t.right(360/8)

def conclusion(w):
    print("Goodbye!")
    w.exitonclick()
```

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

Answer Key:

```
#some comments

def main():
    fileName = input('Enter file name: ')
    infile = open(fileName)
    data = infile.read()
    print("Number of lines:", data.count("\n"))

    infile.close()
```

(b) Write a **complete** program that prints the total area 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, Population, Area (square miles)
Bronx, 1385108, 42
Brooklyn, 2504700, 71
Manhattan, 1585873, 23
Queens, 2230722, 109
Staten Island, 468730, 58
```

Answer Key:

```
#some comments

def main():
    sum = 0
    infile = open("cityData.csv")
    infile.readline()   #Ignore first line, since no numbers
    lines = infile.readlines()
    for l in lines:
        cells = l.split()
        sum = sum + eval(cells[2])

    print("Total population:", sum)

    infile.close()
```

8. Write the Python code for the algorithms below:

   (a)
```
getInput()
Ask user for number between 0 and 100
Until they enter a number between 0 and 100
    Print error message
    Ask user for a number between 0 and 100
Return the number entered
```

   Answer Key:

```
def getInput()
    x = int(eval('Enter a number between 0 and 100: '))
    while not (0 <= x and x<=100):
        print('Error!')
        x = int(eval('Enter a number between 0 and 100: '))
    return(x)
```

   (b)
```
merge(ls, mid)
Initialize the variables: set newList to be an empty list, set counters i to be 0
and j to be mid.
While i < mid and j < len(ls):
    If ls[i] >= ls[j], append ls[i] to the newList and increment i.
```

```
        Else: append ls[j] to the newList and increment j.
    While i < mid:
        Append ls[i] to the newList and increment i.
    While j < len(ls)
        Append ls[j] to the newList and increment j.
    Return newList
```

Answer Key:

```
def merge(ls, mid):
    newList = []
    i, j = 0, mid
    while i < mid and j < len(ls):
        if ls[i] >= ls[j]:
            newList.append(ls[i])
            i += 1
        else:
            newList.append(ls[j])
            j += 1
    while i < mid:
        newList.append(ls[i])
        i += 1
    while j < len(ls)
        newList.append(ls[j])
        j += 1
    Return newList
```

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
```

Answer Key:

```
#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):
    numMoves = 0                                    ###ADDED
    while checkWinner == "No Winner" and numMoves < 9:###ADDED
        numMoves += 1                               ###ADDED
        if numMoves % 2 == 0:                       ###ADDED
            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"
        else:                                       ###ADDED
            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"
    if checkWinner != "No Winner":                  ###ADDED
        print("There was a winner!")                ###ADDED
    else:                                           ###ADDED
        print("Game Over:  No winner!")             ###ADDED
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 number of occurrences (entry after the comma at the end of the line). For example, your function would return the number 83 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
```

Answer Key:

```
#Function that extracts gender from the line:
def getNumber(line):
    words = line.split(",")
    return(eval(line[1]))
```

(b) Write a **complete program** that uses your function above to count the overall number of occurrence. Your program should open the file `NY.txt`, keep a running total of the number of times the names have been used, and print the results.

Answer Key:

```
def main():
    infile = open("NY.txt", "r")
    count = 0
    for line in infile:
        count = count + getNumber(line)
    print("Number of occurrences is:", count)

main()
```