Answer Key: CMP 167 Final Exam, Version 4, Spring 2015

1. What will the following code print:

```
s = "List'(Processing'(John'(McCarthy"
a = s[0:3]
print(a.lower())
names = s.split("'(")
print(names)
b = names[1]
c = names[-1]
print(c,b)
d = a + b[0]
print('(print "', d.upper(),'")')
```

Answer Key:

```
lis
['List', 'Processing', 'John', 'McCarthy']
McCarthy Processing
(print " LISP ")
```

2. Write a **complete program** to calculate how much something will weigh on Saturn. Your program should prompt the user for the weight on the Earth and then print out the weight on Saturn. For example, if the user enters `100`, your program should print out `108`.

   The weight of an item on Saturn is 108% *of its weight on earth.*

Answer Key:

```
#Computes weights on Saturn
def main():
     earthWeight = eval(input('Enter earth weight: '))
     saturnWeight = earthWeight*1.08
     print('The weight on Saturn is:', saturnWeight)

main()
```

3. What is output of the code below:

```
def prob4(washington, adams):
     if washington < 2:
          print("Small case")
          monroe = -1
     else:
          print("Complex case")
          monroe = helper(washington,adams)
     return(monroe)

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

(a)
```
r = prob4(0,"city")
print("Return:  ", r)
```

Output:

Answer Key:

```
Small 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) < 7:
         print("Short Line: ", end ="")
         c = c + 1
      print(line)
   print("Num short lines is", c)

prob5V1()
```

places.txt

```
Greene
Clinton
Warren
Montgomery
Miami
Preble
```

Output:

Answer Key:

```
Short Line: Greene

Clinton

Short Line: Warren

Montgomery

Short Line: Miami

Short Line: Preble

Num short lines is 4
```

5. (a) Write a function that takes number between 1 and 4 as a parameter and returns the corresponding season as a string. For example, if the parameter is 1, your function should return `"winter"`. If the parameter is 2, your function should `"spring"`, etc. If the parameter is not between 1 and 4, your function should return the empty string.

Answer Key:

```
def returnSeason(num):
    if num == 1:
        return "winter"
    elif num == 2:
        return "spring"
    elif num == 3:
        return "summer"
    elif num == 4:
        return "fall"
    else:
        return ""
```

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

6. Complete the following program, which sets up a graphics window and turtle, draws a decagon (10-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()`, `drawDecagon()`, and `conclusion()`:

```
import turtle

def main():
    w,t = setUp()   #sets up a graphics window and turtle
    drawDecagon(t)  #draws a decagon 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 drawDecagon(t):
    for i in range(10):
        t.forward(100)
        t.right(360/10)

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 for cities stored in a data file. Your program should open the file, `cityData.csv` and sum the areas (the area is the last value 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 total that you calculated.

cityData.csv:

```
Borough, County, Area (square miles)
Bronx, Bronx, 42
Brooklyn, Kings, 71
Manhattan, New York, 23
Queens, Queens, 109
Staten Island, Richmond, 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 area:", sum)

    infile.close()
```

8. Write the Python code for the algorithms below:

   (a)
   ```
   getInput()
   Ask user for a string
   Until they enter a non-empty string
       Print error message
       Ask user for a non-empty string
   Return the string entered
   ```

   Answer Key:

   ```
   def getInput()
       s = eval('Enter a string: ')
       while s == "":
           print('Error! Empty String!')
           s = eval('Enter a string: ')
       return(s)
   ```

   (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 smaller than ls[j+1], swap the values
   Return the list, ls.
   ```

Answer Key:

```
def sort(ls):
    L = len(ls)
    for i in range(L-1):
        for j in range(L-1):
            if ls[j] < ls[j+1]:
                ls[j],ls[j+1] = ls[j+1],ls[j]
    return ls
```

9. In lab, we wrote a Tic-Tac-Toe program. Modify the program to check for a winner after each move and keep track of the number of times this occurs. Your program should print out a message if someone has a winning configuration, print out the total winning configurations seen so far, and continue playing. 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):
    numWinners = 0                                  ###ADDED
    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"
        if checkWinner(board):                      ###ADDED
            print('X has a winning configuration!') ###ADDED
            numWinners = numWinners + 1             ###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(board):                      ###ADDED
            print('O has a winning configuration!') ###ADDED
            numWinners = numWinners + 1             ###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"
    if checkWinner(board):                          ###ADDED
        print('X has a winning configuration!')     ###ADDED
        numWinners = numWinners + 1                 ###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) Write a **complete** class that keeps tracks of information about chocolate. Your class, `Chocolate` should contain instance variables for the `name`, `pricePerPound`, `weight` and `countryOfOrigin`, and should have a constructor method as well as a method, `cost()`, that returns the price (`pricePerPound * weight`) for the chocolate and a method, `getWeight()`, that returns the weight for the chocolate.

Answer Key:

```
class Chocolate:
    def __init__(self, name, pricePerPound, weight, countryOfOrigin):
        self.name = name
        self.pricePerPound = pricePerPound
        self.weight = weight
        self.countryOfOrigin = countryOfOrigin

    def cost(self):
        return self.pricePerPound * self.weight

    def getWeight(self):
        return self.weight
```

(b) Write a function that takes as input a list of `chocolate`, called `shoppingList`, and returns the most expensive chocolate in the list (i.e. the maximum of all the costs of the chocolate in the inputted list):

```
    def maxWeight(shoppingList):
```

Answer Key:

```
def maxWeight(shoppingList):
    maxCost = 0
    for c in shoppingList:
        if c.cost() > maxCost:
            maxCost = c.cost()
    return maxCost
```