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

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 1, Spring 2014

1. What will the following code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
print("Two of them are",days[0], days[-1])
result = ""
for i in range(len(days[0])):
    if i > 2:
        result = result + days[0][i]
print("My favorite", result, "is Saturday.")
```

2. Define a Python function named `calculate_tax` which accepts one parameter, `income`, and returns the income tax. Income is taxed according to the following rule: the first $200,000 is taxed at 25% and any remaining income is taxed at 50%. For example, `calculate_tax(100000)` should return $100,000 \times 0.25 = 25,000$, while `calculate_tax(300000)` should return $200,000 \times 0.25 + 100,000 \times 0.5 = 100,000$.

3. Complete the following program. That is, write the functions `getInputs()`, `countWord()`, `average()`, and `printSummary()`:

```
def main():
    fname, word = getInputs()   #get the file name and word to be searched
    infile = open(fname, "r")   #open the file for reading
    resultList = list()         #initialize result list to empty list

    for line in infile:
        num = countWord(line, word) #return the number of
                                    #times word occurs in line
        resultList.append(num)

    a = average(resultList)     #compute the average number of
                                #times word occurs per line
    printSummary(word, a)       #print the average (including explanation)
```

4. Given the following function definitions:

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

(a) What does `foo([1,2,3,4])` return?

(b) What does `foo([1024,512,256,128])` return?

5. Given the following code:

```
file = open("numbers.txt")
total = 0
for line in file.readlines():
    for strnum in line.split(","):
        num = int(strnum)
        if num % 2 == 0:
            total = total + num
            print(total)
```

(a) What will the output be for this `numbers.txt`?
numbers.txt:

```
1,2,3,4,5,6
```

(b) What will the output be for this `numbers.txt`?
numbers.txt:

```
123456
```

6. Draw what will be displayed in the graphics window when the following program is executed. Remember to indicate the final position and direction of the turtle at the end of program. (The turtle always points to the right of the screen at the start of the program.)

```
from turtle import *

def mystery(t, n, d):
    for i in range(n):
        if d == 'r':
            t.right(360/n)
        else:
            t.left(360/n)
        t.forward(50)
```

Graphics Displayed:

```
def draw(t, n):
    t.forward(100)
    mystery(t, n, 'r')
    mystery(t, n, 'l')


t = Turtle()
draw(t, 4)
```

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the lines containing the phrase: `The Amazing Spider Man` (that is, the line must contain all four words in this order):

8. Write the python code for the algorithms below:

(a)
```
find(st)
        set index to 0
        set location to -1
        set found to false
        while not found
            if st[index] equals ','
                set location to index
                set found to true
            increment index
        return location
```

(b)
```
getSmaller(ls)
        for each item in ls
            if current item is less than first item in ls
                switch first item and current item in ls
```

9. In the book, a racquetball program was designed. Modify the design to simulate games of another racquet sport, squash. Amateur squash scoring rules are slightly different than racquetball: if a player wins the rally (whether or not they were serving), that player earns a point ("point-a-rally" scoring (PARS)). (As in racquetball, if the player loses the rally, the player loses the serve.) The first player whose score is 11 and above and who is ahead by 2 wins. For example, if the score is 11-4, player A would win. But if the score is 11-10, play continues until one player is ahead by two.

   Clearly mark your changes to the design below to create a squash simulation program:

```
# rball.py
from random import random
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
def simNGames(n, probA, probB):
    # Simulates n games of racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns number of wins for A and B
    winsA = winsB = 0
    for i in range(n):
        scoreA, scoreB = simOneGame(probA, probB)
        if scoreA > scoreB:
            winsA = winsA + 1
        else:
            winsB = winsB + 1
    return winsA, winsB
def simOneGame(probA, probB):
    # Simulates a single game or racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns final scores for A and B
    serving = "A"
    scoreA = 0
    scoreB = 0
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
    return scoreA, scoreB
def gameOver(a, b):
    # a and b represent scores for a racquetball game
    # Returns True if the game is over, False otherwise.
    return a==15 or b==15
```

10. (a) Write a **complete** class that keeps tracks of information about songs. Your class, `Song` should contain instance variables for the `name`, `length`, `artist` and `composer`, and should have a constructor method as well as a method that returns the length of the song.

(b) Write a function that takes as input a list of `Song`s, called `mixTape`, and returns the sum of the lengths of the songs in the list:

```
def tapeLength(mixTape):
```

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.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

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

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

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

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 2, Spring 2014

1. What will the following code print:

```
s = "marchxoctoberxjanuaryxaugustx"
num = s.count("x")
items = s[:-1].split("x")
result = ""
for item in items:
    print( item.capitalize() )
    result = result + item[0].upper()
print( (result[0:2] + "NTHS!! ") * 3, end="")
```

2. Define a Python function named `calculate_tax` which accepts one parameter, `income`, and returns the income tax. Income is taxed according to the following rule: the first $100,000 is taxed at 25% and any remaining income is taxed at 50%. For example, `calculate_tax(80000)` should return $80,000 \times 0.25 = 20,000$, while `calculate_tax(200000)` should return $100,000 \times 0.25 + 100,000 \times 0.5 = 75,000$.

3. Complete the following program that is, write the functions `getInputs()`, `countAs()`, `average(l)`, and `printSummary(a)`:

```
def main():
    fname = getInputs()          #get the file name
    infile = open(fname, "r")    #open the file for reading
    resultList = list()          #initialize result list to empty list

    for line in infile:
        num = countAs(line)      #return the number of 'a' and 'A' in line
        resultList.append(num)

    a = average(resultList)      #compute the average number of
                                 #times 'a' or 'A' occurs per line
    printSummary(a)              #print the average (including explanation)
```

4. Given the following function definitions:

```
def bar(n):
    if n >= 32:
        return 2
    else:
        return 1

def foo(l):
    n = bar(l[2])
    return l[n]
```

(a) What does `foo([1,2,3,4])` return?

(b) What does `foo([1024,512,256,128])` return?

5. Given the following code:

```
file = open("numbers.txt")
total = 0
for line in file.readlines():
    for strnum in line.split(","):
        num = int(strnum)
        if num % 2 == 0:
            total = total + num
            print(total)
```

(a) What will the output be for this `numbers.txt`?
numbers.txt:

```
10,11,12,13,14
```

(b) What will the output be for this `numbers.txt`?
numbers.txt:

```
1011121314
```

6. Draw what would be displayed in the graphics window when the following program is executed. Remember to indicate the final position and direction of the turtle at the end of program. (The turtle always points to the right of the screen at the start of the program.)

```
from turtle import *

def mystery(t, n, d):
    for i in range(n):
        if d == 'r':
            t.right(360/n)
        else:
            t.left(360/n)
        t.forward(50)

def draw(t, n):
    t.forward(100)
    mystery(t, n, 'r')
    mystery(t, n, 'l')


t = Turtle()
draw(t, 3)
```

Graphics Displayed:

7. Write a **program** that reads in a text file, `infile.txt`, and replace each line with the word `Awesome` (that is, every line of the `infile.txt` should be `Awesome`), then prints out the total number of lines in the file.

8. Write the python code for the algorithms below:

   (a)
```
find(st)
        set index to (length of st) - 1
        set location to -1
        set found to false
        while not found
            if st[index] equals ','
                set location to index
                set found to True
            decrement index
        return location
```

   (b)
```
getBigger(ls)
        for each item in ls
            if current item is greater than first item in ls
                switch first item and current item in ls
```

9. In the book, a racquetball program was designed. Modify the design to simulate games of volleyball. Volleyball scoring rules are slightly different than racquetball: if a player wins the rally (whether or not they were serving), that player earns a point ("point-a-rally" scoring (PARS)). (As in racquetball, if the player loses the rally, the player loses the serve.) The first player whose score is 25 and above and who is ahead by 2 wins. For example, if the score is 25-4, player A would win. But if the score is 25-24, play continues until one player is ahead by two.

   Clearly mark your changes to the design below to create a volleyball simulation program:

```
# rball.py
from random import random
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
def simNGames(n, probA, probB):
    # Simulates n games of racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns number of wins for A and B
    winsA = winsB = 0
    for i in range(n):
        scoreA, scoreB = simOneGame(probA, probB)
        if scoreA > scoreB:
            winsA = winsA + 1
        else:
            winsB = winsB + 1
    return winsA, winsB
def simOneGame(probA, probB):
    # Simulates a single game or racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns final scores for A and B
    serving = "A"
    scoreA = 0
    scoreB = 0
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
    return scoreA, scoreB
def gameOver(a, b):
    # a and b represent scores for a racquetball game
    # Returns True if the game is over, False otherwise.
    return a==15 or b==15
```

10. (a) Write a **complete** class that keeps tracks of information about movies. Your class, `Movie` should contain instance variables for the `name`, `length`, `studio` and `director`, and should have a constructor method as well as a method that returns the length of the movie.

(b) Write a function that takes as input a list of `Movie`s, called `driveContents` and returns the sum of the lengths of the movies in the list:

```
def viewLength(driveContents):
```

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.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

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

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

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

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 3, Spring 2014

1. What will the following code print:

```
s = "history.biology.french.trigonometry.science."
num = s.count(".")
subjects = s[:-1].split(".")
print("There are", num, "important subjects in school.")
for item in subjects[:-1]:
    print("Don't know much about", item)
print("But I do know that I love computer " + subjects[4])
```

2. Define a Python function named `calculate_tax` which accepts one parameter, `income`, and returns the income tax. Income is taxed according to the following rule: the first \$50,000 is taxed at 10% and any remaining income is taxed at 20%. For example, `calculate_tax(40000)` should return $40,000 \times 0.1 = 4,000$, while `calculate_tax(100000)` should return $50,000 \times 0.1 + 50,000 \times 0.2 = 15,000$.

3. Complete the following program that is, write the functions `getInputs()`, `countSpaces()`, `minMax()`, and `printSummary()`:

```
def main():
    fname = getInputs()         #get the file name
    infile = open(fname, "r")   #open the file for reading
    resultList = list()         #initialize result list to empty list

    for line in infile:
        num = countSpaces(line) #return the number of spaces in line
        resultList.append(num)

    m,M = minMax(resultList)     #compute the minimum and maximum spaces per line
    printSummary(m,M)       #print the min and max spaces (including explanation)
```

4. Given the following function definitions:

```
def bar(n):
    if n < 8:
        return -1
    else:
        return n//2

def foo(l):
    n = bar(l[3])
    return 2*n
```

(a) What does `foo([1,2,3,4])` return?

(b) What does `foo([1024,512,256,128])` return?

5. Given the following code:

```
file = open("numbers.txt")
total = 0
for line in file.readlines():
    for strnum in line.split(","):
        num = int(strnum)
        if num % 2 == 0:
            print(num)
            total = total + num
print(total)
```

(a) What will the output be for this `numbers.txt`?
numbers.txt:

```
1,2,3,4,5,6
```

(b) What will the output be for this `numbers.txt`?
numbers.txt:

```
123456
```

6. Draw what would be displayed in the graphics window when the following program is executed. Remember to indicate the final position and direction of the turtle at the end of program. (The turtle always points to the right of the screen at the start of the program.)

```
from turtle import *

def mystery(t, n, d):
    for i in range(n):
        if d == 'r':
            t.right(360/n)
        else:
            t.left(360/n)
        t.forward(50)
```

Graphics Displayed:

```
def draw(t, n):
    t.backward(100)
    mystery(t, n, 'l')
    mystery(t, n, 'r')


t = Turtle()
draw(t, 3)
```

7. Write a **program** that reads in a text file, `infile.txt`, and prints out each line surrounded by `'-*-'`.

8. Write the python code for the algorithms below:

   (a) `find(st)`

```
set index to 0
set location to -1
set firstFound to false
set notFound to true
while notFound and index < length st
    if st[index] equals ',' and firstFound is false
        set firstFound to true
    otherwise, if st[index] equals ','
        set location to index
        set notFound to false
    increment index
return location
```

   (b) `getBigger(ls)`

```
for each item in ls
    if current item is greater than last item in ls
        switch last item and current item in ls
```

9. In the book, a racquetball program was designed. Modify the design to simulate games of another racquet sport, badminton. Badminton scoring rules are slightly different than racquetball: if a player wins the rally (whether or not they were serving), that player earns a point ("point-a-rally" scoring (PARS)). (As in racquetball, if the player loses the rally, the player loses the serve.) The first player whose score is 21 and above and who is ahead by 2 wins. For example, if the score is 21-4, player A would win. But if the score is 21-20, play continues until one player is ahead by two.

   Clearly mark your changes to the design below to create a badminton simulation program:

```
# rball.py
from random import random
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
def simNGames(n, probA, probB):
    # Simulates n games of racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns number of wins for A and B
    winsA = winsB = 0
    for i in range(n):
        scoreA, scoreB = simOneGame(probA, probB)
        if scoreA > scoreB:
            winsA = winsA + 1
        else:
            winsB = winsB + 1
    return winsA, winsB
def simOneGame(probA, probB):
    # Simulates a single game or racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns final scores for A and B
    serving = "A"
    scoreA = 0
    scoreB = 0
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
    return scoreA, scoreB
def gameOver(a, b):
    # a and b represent scores for a racquetball game
    # Returns True if the game is over, False otherwise.
    return a==15 or b==15
```

10. (a) Write a **complete** class that keeps tracks of information about books. Your class, `Book`, should contain instance variables for the `title`, `length`, `author` and `publisher`, and should have a constructor method as well as a method that returns the length of the book.

(b) Write a function that takes as input a list of `Book`, called `library` and returns the sum of the lengths of the books in the list:

```
def libraryPages(library):
```

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.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

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

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

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

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 4, Spring 2014

1. What will the following code print:

```
s = "omelettesporridgescerealspancakes"
num = s.count("s")
breakfast = s[:-1].split("s")
print("You have a choice of", num, "options:")
for item in breakfast:
    print(item.capitalize())
print("\nBut I need " + breakfast[0][1] + breakfast[1][1] + breakfast[2][2:4] + "!!!")
```

2. Define a Python function named `calculate_tax` which accepts one parameter, `income`, and returns the income tax. Income is taxed according to the following rule: the first \$500,000 is taxed at 50% and any remaining income is taxed at 75%. For example, `calculate_tax(400000)` should return $400,000 \times 0.5 = 200,000$, while `calculate_tax(600000)` should return $500,000 \times 0.5 + 100,000 \times 0.75 = 325,000$.

3. Complete the following program that is, write the functions `getInputs()`, `countSpaces()`, `calculate()`, and `printSummary()`:

```
def main():
    fname = getInputs()         #get the file name
    infile = open(fname, "r")   #open the file for reading
    resultList = list()         #initialize result list to empty list

    for line in infile:
        num = countSpaces(line) #return the number of spaces in line
        resultList.append(num)

    n = calculate(resultList)  #compute number of lines with more than 5 spaces
    printSummary(n)       #print the number of long lines (including explanation)
```

4. Given the following function definitions:

```
def bar(n):
    if n >= 8:
        return 8
    else:
        return n*2

def foo(l):
    n = bar(l[1])
    return n//2
```

(a) What does `foo([1,2,3,4])` return?

(b) What does `foo([1024,512,256,128])` return?

5. Given the following code:

```
file = open("numbers.txt")
total = 0
for line in file.readlines():
    for strnum in line.split(","):
        num = int(strnum)
        if num % 2 == 0:
            print(num)
            total = total + num
print(total)
```

(a) What will the output be for this `numbers.txt`?
numbers.txt:

```
5,6,7,8,9
```

(b) What will the output be for this `numbers.txt`?
numbers.txt:

```
5
6
7
8
9
```

6. Draw what would be displayed in the graphics window when the following program is executed. Remember to indicate the final position and direction of the turtle at the end of program. (The turtle always points to the right of the screen at the start of the program.)

```
from turtle import *

def mystery(t, n, d):
    for i in range(n):
        if d == 'r':
            t.right(360/n)
        else:
            t.left(360/n)
        t.forward(50)

def draw(t, n):
    t.backward(100)
    mystery(t, n, 'l')
    mystery(t, n, 'r')


t = Turtle()
draw(t, 4)
```

Graphics Displayed:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out each line uppercase except for first character on each line. For example, "Hello World" should be printed out as "hELLO WORLD".

8. Write the python code for the algorithms below:

   (a)
```
find(st)
        set index to (length of st) - 1
        set location to -1
        set firstFound to false
        set notFound to true
        while notFound and index > -1
            if st[index] equals ',' and firstFound is false
                set firstFound to true
            otherwise, if st[index] equals ','
                set location to index
                set notFound to false
            decrement index
        return location
```

   (b)
```
getSmaller(ls)
        for each item in ls
            if current item is smaller than last item in ls
                switch last item and current item in ls
```

9. In the book, a racquetball program was designed. Modify the design to simulate games of another racquet sport, table tennis ("ping pong"). Table tennis scoring rules are slightly different than racquetball: if a player wins the rally (whether or not they were serving), that player earns a point ("point-a-rally" scoring (PARS)). (As in racquetball, if the player loses the rally, the player loses the serve.) The first player whose score is 11 and above and who is ahead by 2 wins. For example, if the score is 11-4, player A would win. But if the score is 11-10, play continues until one player is ahead by two.

   Clearly mark your changes to the design below to create a table tennis simulation program:

```
# rball.py
from random import random
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
def simNGames(n, probA, probB):
    # Simulates n games of racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns number of wins for A and B
    winsA = winsB = 0
    for i in range(n):
        scoreA, scoreB = simOneGame(probA, probB)
        if scoreA > scoreB:
            winsA = winsA + 1
        else:
            winsB = winsB + 1
    return winsA, winsB
def simOneGame(probA, probB):
    # Simulates a single game or racquetball between players whose
    #    abilities are represented by the probability of winning a serve.
    # Returns final scores for A and B
    serving = "A"
    scoreA = 0
    scoreB = 0
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
    return scoreA, scoreB
def gameOver(a, b):
    # a and b represent scores for a racquetball game
    # Returns True if the game is over, False otherwise.
    return a==15 or b==15
```

10. (a) Write a **complete** class that keeps tracks of information about Olympic athletes. Your class, `Athlete` should contain instance variables for the `name`, `numberOfMedals`, `country` and `sport`, and should have a constructor method as well as a method that returns the number of medals for the athlete.

(b) Write a function that takes as input a list of `Athlete`s, called `team`, and returns the sum of the number of the medals in the list:

```
def overallMedalCount(team):
```

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.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

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

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving