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

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

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

1. What will the following code print:

```
a = ",Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec,"
b = "Apr 15, 2014"
c = b.split()
print(c)
d = a.split(",")
print(d[1:12])
e = (a.find(c[0]) - 1) / 3
print(e)
f = c[1][:-1]
print(str(int(e)) + "/" + f + "/" + c[2])
```

2. Write a program to print the fine for speeding. The program must read the speed from user input, then compute and print the fine. The fine is $12 for each mph over 65 and less than or equal to 70, and $15 for each additional mph over 70.

   For example, if the speed is 68 mph, then the fine would be $36 = $12 x 3. If the speed is 72 mph, then the fine would be $90 = $12 x 5 + $15 x 2.

3. Complete the following program, which reads in a file that has multiple grades, each separated by a comma, and prints out the computed average. That is, write the functions `getGrades()` and `calculateAverage()`:

```
def main():
    grades = getGrades()   #get the file name containing the grades
                           #and return the contents of the file
    avg = calculateAverage(grades) #separate the grades into numbers and compute
                           #the average
    print("The calculated average is:", avg)

main()
```

4. Given the following function definitions:

```
def help(g):
    s = 1
    for h in g:
        s = s + h
        print(s)
    return s

def abc(d):
    e = len(d)
    print("e is ", e)
    if e >= 2:
        f = help(d[0:3])
    elif 2 > e >= 1:
        f = help(d[0:1])
    else:
        f = 5
    return f
```

(a) What does `abc([0,1,2,3])` return?

Write output for partial credit:

(b) What does `abc([49])` return?

Write output for partial credit:

5. Given the following code:

```
def main():
    file = open("poetry.txt", 'r')
    count = 0
    for line in file:
        line2 = line[:-1] + "?"
        if count % 2 == 0:
            print(line2)
        else:
            print(len(line[:-1]))
        count = count + 1

main()
```

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

poetry.txt:

```
What a
nice
day.
It is.
```

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

poetry.txt:

```
No rain
but
cloudy.
```

6. (a) 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):

    for i in range(n):
        t.backward(50)
        if i % 2 == 0:
            t.right(90)
        else:
           t.left(90)

def draw(t, n):
    mystery(t, n)

t = Turtle()
draw(t, 5)
```

Graphics Displayed:

(b) Write a complete program that asks the user for the radius and color of a circle, creates a graphics window, and draws a circle based on the inputted information. Your `main()` should use functions to complete these tasks, that is, in addition to the `main()` you should write the additional functions `getInput()`, `createWin()`, and `draw()`:

7. Write a program that reads in a file called **infile.txt**. For each line in the file it should print out the line followed by "- * -" and then the number of times the lower-case word *the* appears in that line. Finally, it should print out the total number of *the*'s in the file.

8. Write the Python code for the algorithms below:

   (a)
```
count(ls)
        Set count to 0
        for each item in the list ls
            If item is negative
                increment count
        print count
```

   (b)
```
search(ls, key, first, last)
        while first is less than last
            Set mid to first + last / 2
            if ls[mid] equals key
                return mid
            else if ls[mid] < key
                first = mid + 1
            else
                last = mid -1
        return -1
```

9. (a) Write a **complete** class that keeps tracks of information about countries. Your class, `Country` should contain instance variables for the `name`, `population`, `area` and `chocolateProduction`, and should have a constructor method as well as a method, `populationDensity()`, that returns the population density (``population/area'') for the country and a method, `getChocolateProduction()`, that returns the chocolate production for the country.

(b) Write a function that takes as input a list of `Countries`, called `continent`, and returns the sum of the chocolate production in the list:

```
def overallChocolateProduction(continent):
```

10. In the book, a `Student` class and program for finding the student with the highest GPA was designed. Modify the design to also find all full-time high honor students (those with a GPA ≥ 3.75 and currently enrolled in ≥ 12 credits). Your program should print out the name and GPA of all students who meet this criteria. The current credits will be provided as the last entry on each line of the file. Include in your class definition, an instance variable to hold current credits and a new method to access it.

    Clearly mark your changes to the design below:

```
class Student:
    def __init__(self, name, hours, qpoints):
        self.name = name
        self.hours = float(hours)
        self.qpoints = float(qpoints)
    def getName(self):
        return self.name
    def getHours(self):
        return self.hours
    def getQPoints(self):
        return self.qpoints
    def gpa(self):
        return self.qpoints/self.hours

def makeStudent(infoStr):
    # infoStr is a tab-separated line: name hours qpoints
    # returns a corresponding Student object
    name, hours, qpoints = infoStr.split("\t")
    return Student(name, hours, qpoints)

def main():
    # open the input file for reading
    filename = input("Enter name the grade file: ")
    infile = open(filename, 'r')
    # set best to the record for the first student in the file
    best = makeStudent(infile.readline())
    # process subsequent lines of the file
    for line in infile:
        # turn the line into a student record
        s = makeStudent(line)
        # if this student is best so far, remember it.
        if s.gpa() > best.gpa():
            best = s
    infile.close()
    # print information about the best student
    print("The best student is:", best.getName())
    print("hours:", best.getHours())
    print("GPA:", best.gpa())
```

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 11-1 TTH 9-11 TTH 11-1
TTH 1-3 TTH 6 - 8

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

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

1. What will the following code print:

```
a = ",Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec,"
b = "Mar 15, 2014"
c = b.split()
print(c)
d = a.split(",")
print(d[1:12])
e = a.find(c[0]) / 3
print(e)
f = c[1][:-1]
print(str(int(e)) + "/" + f + "/" + c[2])
```

2. Write a program to print the fine for speeding. The program must read the speed from user input, then compute and print the fine. The fine is $10 for each mph over 55 and less than or equal to 65, and $15 for each additional mph over 65.

   For example, if the speed is 58 mph, then the fine would be $30 = $10 x 3. If the speed is 67 mph, then the fine would be $130 = $10 x 10 + $15 x 2.

3. Complete the following program, which reads in a file that has multiple grades, each separated by a semi-colon, and prints out the computed average. That is, write the functions `retrieveGrades()` and `computeAverage()`:

```
def main():
    grades = retrieveGrades()    #get the file name containing the grades
                                 #and return the contents of the file
    avg = computeAverage(grades) #separate the grades into numbers and compute
                                 #the average
    print("The calculated average is:", avg)

main()
```

4. Given the following function definitions:

```
def help(g):
    s = 0
    for h in g:
        s = s + 2
        print(s)
    return s

def abc(d):
    e = len(d) - 1
    print("e is", e)
    if e >= 3:
        f = help(d[0:2])
    elif 2 >= e >= 1:
        f = help(d[0:1])
    else:
        f = 10
    return f
```

(a) What does `abc([7,8,9])` return?

Write output for partial credit:

(b) What does `abc([77])` return?
Write output for partial credit:

5. Given the following code:

```
def main():
    file = open("summary.txt", 'r')
    count = 0
    for line in file:
        line2 = line[:-1] + "?"
        if count % 2 == 0:
            print("count", line2)
        else:
            print(count, line2)
        count = count + 1

main()
```

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

```
It was
short.
```

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

```
So
so
short.
```

6. (a) 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):

    for i in range(n):
        t.backward(50)
        if i % 2 == 0:
            t.left(90)
        else:
           t.right(90)

def draw(t, n):
    mystery(t, n)

t = Turtle()
draw(t, 5)
```

Graphics Displayed:

(b) Write a complete program that asks the user for the outline and fill color of a circle, creates a graphics window, and draws a circle with those colors centered in the window with radius 50. Your `main()` should use functions to complete these tasks, that is, in addition to the `main()` you should write the additional functions `getInput()`, `createWin()`, and `draw()`:

7. Write a program that reads in a file called **infile.txt**. For each line in the file, the program should print out "- * -" to an output file called **outfile.txt**. Finally, it should print the total number of lines in the input file to the screen.

8. Write the Python code for the algorithms below:

   (a)
```
count(ls)
        Set count to 0
        for each item in the list ls
            If item is positive
                increment count
        print count
```

   (b)
```
search(ls, key, first, last)
        while first is less than last
            Set mid to first + last / 2
            if ls[mid] is less than key
                Set first to mid + 1
            else
                Set last to mid
        if last equals first and ls[first] equals key
            return first
        else
            return -1
```

9. (a) Write a **complete** class that keeps tracks of information about states. Your class, `State` should contain instance variables for the `name`, `numberOfReps`, `population` and `area`, and should have a constructor method as well as a method, `getNumReps()`, that returns the number of representatives for the state and a method, `populationDensity()` that calculates population density (``population/area''):

(b) Write a function that takes as input a list of `State`s, called `region`, and returns the sum of the number of the representatives for the states in the list:

```
def overallNumReps(region):
```

10. In the book, a `Student` class and program for finding the student with the highest GPA was designed. Modify the design to also find all part-time high honor students (those with a GPA ≥ 3.75 and currently enrolled in < 12 credits). Your program should print out the name and GPA of all students who meet this criteria. The current credits will be provided as the last entry on each line of the file. Include in your class definition, an instance variable to hold current credits and a new method to access it.

    Clearly mark your changes to the design below:

```
class Student:
    def __init__(self, name, hours, qpoints):
        self.name = name
        self.hours = float(hours)
        self.qpoints = float(qpoints)
    def getName(self):
        return self.name
    def getHours(self):
        return self.hours
    def getQPoints(self):
        return self.qpoints
    def gpa(self):
        return self.qpoints/self.hours

def makeStudent(infoStr):
    # infoStr is a tab-separated line: name hours qpoints
    # returns a corresponding Student object
    name, hours, qpoints = infoStr.split("\t")
    return Student(name, hours, qpoints)

def main():
    # open the input file for reading
    filename = input("Enter name the grade file: ")
    infile = open(filename, 'r')
    # set best to the record for the first student in the file
    best = makeStudent(infile.readline())
    # process subsequent lines of the file
    for line in infile:
        # turn the line into a student record
        s = makeStudent(line)
        # if this student is best so far, remember it.
        if s.gpa() > best.gpa():
            best = s
    infile.close()
    # print information about the best student
    print("The best student is:", best.getName())
    print("hours:", best.getHours())
    print("GPA:", best.gpa())
```

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 11-1 TTH 9-11 TTH 11-1
TTH 1-3 TTH 6-8

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

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

1. What will the following code print:

```
a = ",Dec,Nov,Oct,Sep,Aug,Jul,Jun,May,Apr,Mar,Feb,Jan,"
b = "Nov 15, 2014"
c = b.split()
print(c)
d = a.split(",")
print(d[1:12])
e = (a.find(c[0]) - 1) / 4 + 1
print(e)
f = c[1][:-1]
print(str(int(e)) + "/" + f + "/" + c[2])
```

2. Write a program to print the fine for speeding. The program must read the speed from user input, then compute and print the fine. The fine is $15 for each mph over 60 and less than or equal to 70, and $20 for each additional mph over 70.

   For example, if the speed is 63 mph, then the fine would be $45 = $15 x 3. If the speed is 72 mph, then the fine would be $190 = $15 x 10 + $20 x 2.

3. Complete the following program, which reads in a file that has multiple grades, each separated by a colon, and prints out the computed average. That is, write the functions `extractGrades()` and `processAverage()`:

```
def main():
    grades = extractGrades() #get the file name containing the grades
                             #and return the contents of the file
    avg = processAverage(grades) #separate the grades into numbers and compute
                             #the average
    print("The calculated average is:", avg)

main()
```

4. Given the following function definitions:

```
def help(g):
    s = 1
    for h in g:
        s = s + 1
        print(s)
    return s

def abc(d):
    e = len(d)
    print("e is ", e)
    if 5 > e > 2:
        f = help(d[0:3])
    elif e > 5:
        f = help(d[2:5])
    else:
        f = 8
    return f
```

(a) What does `abc([10,20,30,40,50,60])` return?

Write output for partial credit:

(b) What does `abc([5,6,7])` return?

Write output for partial credit:

5. Given the following code:

```
def main():
    file = open("story.txt", 'r')
    count = 0
    for line in file:
        line2 = "!" + line[:-1]
        if count == 2:
            print(line2)
        else:
            print(line.count("a"))
        count = count + 2

main()
```

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

story.txt:

```
Once
upon a
time.
```

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

story.txt:

```
Here
is
a
story...
```

6. (a) 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.)

Graphics Displayed:

```
from turtle import *

def mystery(t, n):

    for i in range(n):
        t.right(90)
        if i % 2 == 0:
            t.backward(50)
        else:
           t.forward(50)

def draw(t, n):
    mystery(t, n)

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

(b) Write a complete program that asks the user for the width and height of a rectangle, creates a graphics window, and draws a rectangle with its upper left corner at $(0, 0)$ and the inputted height and width. Your `main()` should use functions to complete these tasks, that is, in addition to the `main()` you should write the additional functions `getInput()`, `createWin()`, and `draw()`:

7. Write a program that reads in a file called **infile.txt**. For each line in the file it should print out the line followed by "- * -" and then the number of times that the lower-case word *the* appears in that line. Finally, it should print out the total number of times that the lower-case word *the* appears in the file.

8. Write the Python code for the algorithms below:

   (a)
```
balance(ls)
        Set balance to 1000
        for each item in the list ls
            Subtract item from balance
        print balance
```

   (b)
```
search(ls, key, first, last)
        while first is less than last
            Set mid to first + last / 2
            if ls[mid] equals key
                return mid
            else if ls[mid] < key
                first = mid + 1
            else
                last = mid -1
        return -1
```

9. (a) Write a **complete** class that keeps tracks of information about boroughs. Your class, `Borough` should contain instance variables for the `name`, `collegeCampuses`, `population` and `area`, and should have a constructor method as well as a method, `getNumCampuses()`, that returns the number of college campuses and a method, `populationDensity()` that calculates population density (``population/area'') for the borough:

(b) Write a function that takes as input a list of `Boroughs`s, called `city`, and returns the sum of the number of the college campuses for the boroughs in the list:

```
def overallCollegeCampuses(city):
```

10. In the book, a `Student` class and program for finding the student with the highest GPA was designed. Modify the design to also find all part-time honor students (those with a GPA ≥ 3.5 and currently enrolled in < 12 credits). Your program should print out the name and GPA of all students who meet this criteria. The current credits will be provided as the last entry on each line of the file. Include in your class definition, an instance variable to hold current credits and a new method to access it.

    Clearly mark your changes to the design below:

```
class Student:
    def __init__(self, name, hours, qpoints):
        self.name = name
        self.hours = float(hours)
        self.qpoints = float(qpoints)
    def getName(self):
        return self.name
    def getHours(self):
        return self.hours
    def getQPoints(self):
        return self.qpoints
    def gpa(self):
        return self.qpoints/self.hours

def makeStudent(infoStr):
    # infoStr is a tab-separated line: name hours qpoints
    # returns a corresponding Student object
    name, hours, qpoints = infoStr.split("\t")
    return Student(name, hours, qpoints)

def main():
    # open the input file for reading
    filename = input("Enter name the grade file: ")
    infile = open(filename, 'r')
    # set best to the record for the first student in the file
    best = makeStudent(infile.readline())
    # process subsequent lines of the file
    for line in infile:
        # turn the line into a student record
        s = makeStudent(line)
        # if this student is best so far, remember it.
        if s.gpa() > best.gpa():
            best = s
    infile.close()
    # print information about the best student
    print("The best student is:", best.getName())
    print("hours:", best.getHours())
    print("GPA:", best.gpa())
```

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 11-1 TTH 9-11 TTH 11-1
TTH 1-3 TTH 6-8

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

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

1. What will the following code print:

```
a = ",Dec,Nov,Oct,Sep,Aug,Jul,Jun,May,Apr,Mar,Feb,Jan,"
b = "Oct 15, 2014"
c = b.split()
print(c)
d = a.split(",")
print(d[1:12])
e = a.find(c[0]) / 3
print(e)
f = c[1].rstrip(",")
print(str(int(e)) + "/" + f + "/" + c[2])
```

2. Write a program to print the fine for speeding. The program must read the speed from user input, then compute and print the fine. The fine is $15 for each mph over 45 and less than or equal to 65, and $25 for each additional mph over 65.

   For example, if the speed is 48 mph, then the fine would be $45 = $15 x 3. If the speed is 67 mph, then the fine would be $350 = $15 x 20 + $25 x 2.

3. Complete the following program, which reads in a file that has multiple grades, each separated by a hyphen, and prints out the computed average. That is, write the functions `acquireGrades()` and `determineAverage()`:

```
def main():
    grades = acquireGrades() #get the file name containing the grades
                             #and return the contents of the file
    avg = determineAverage(grades) #separate the grades into numbers and compute
                             #the average
    print("The calculated average is:", avg)

main()
```

4. Given the following function definitions:

```
def help(g):
    s = 0
    for h in g:
        s = s + h
        print(s)
    return s

def abc(d):
    e = len(d) + 1
    print("e is ", e)
    if e >= 3:
        f = help(d[0:2])
    elif 3 > e >= 2:
        f = help(d[0:1])
    else:
        f = 65
    return f
```

(a) What does `abc([2,2])` return?

Write output for partial credit:

(b) What does `abc([99])` return?

Write output for partial credit:

5. Given the following code:

```
def main():
    file = open("poetry.txt", 'r')
    count = 0
    for line in file:
        line2 = line[:-1] + "?"
        if count % 2 == 0:
            print(line2)
        else:
            print(len(line[:-1]))
        count = count + 1

main()
```

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

poetry.txt:

```
What a
nice
day.
It is.
```

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

poetry.txt:

```
No rain
but
cloudy.
```

6. (a) 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):

    for i in range(n):
        t.right(90)
        if i % 2 == 0:
            t.backward(50)
        else:
           t.right(90)
           t.forward(50)

def draw(t, n):
    mystery(t, n)

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

Graphics Displayed:

(b) Write a complete program that asks the user for the width and color of a square, creates a graphics window, and draws a square with its lower right corner at $(199, 199)$ and width and color based on the inputted information. Your `main()` should use functions to complete these tasks, that is, in addition to the `main()` you should write the additional functions `getInput()`, `createWin()`, and `draw()`:

7. Write a program that reads in a file called **infile.txt**. For each line in the file, the program should print out the line followed by ": " and then the number of times that the lower-case letter *a* appears in that line. Finally, it should print out the total number of times the lower-case letter *a* appear in the file.

8. Write the Python code for the algorithms below:

   (a)
```
total(ls)
        Set total to 0
        for each item in the list ls
            Add item to total
        print total
```

   (b)
```
search(ls, key, first, last)
        while first is less than last
            Set mid to first + last / 2
            if ls[mid] is less than key
                Set first to mid + 1
            else
                Set last to mid
        if last equals first and ls[first] equals key
            return first
        else
            return -1
```

9. (a) Write a **complete** class that keeps tracks of information about train lines. Your class, `TrainLine` should contain instance variables for the `name`, `length`, `dailyRidership` and `coverageArea`, and should have a constructor method as well as a method, `getLength()`, that returns the train length a method, `riderDensity()` that calculates rider density (``dailyRidership/coverageArea''):

(b) Write a function that takes as input a list of `TrainLine`s, called `subway`, and returns the sum of the length of the train lines in the list:

```
def overallLength(subway):
```

10. In the book, a `Student` class and program for finding the student with the highest GPA was designed. Modify the design to also find all full-time honor students (those with a GPA $\geq$ 3.5 and currently enrolled in $\geq$ 12 credits). Your program should print out the name and GPA of all students who meet this criteria. The current credits will be provided as the last entry on each line of the file. Include in your class definition, an instance variable to hold current credits and a new method to access it.

    Clearly mark your changes to the design below:

```
class Student:
    def __init__(self, name, hours, qpoints):
        self.name = name
        self.hours = float(hours)
        self.qpoints = float(qpoints)
    def getName(self):
        return self.name
    def getHours(self):
        return self.hours
    def getQPoints(self):
        return self.qpoints
    def gpa(self):
        return self.qpoints/self.hours

def makeStudent(infoStr):
    # infoStr is a tab-separated line: name hours qpoints
    # returns a corresponding Student object
    name, hours, qpoints = infoStr.split("\t")
    return Student(name, hours, qpoints)

def main():
    # open the input file for reading
    filename = input("Enter name the grade file: ")
    infile = open(filename, 'r')
    # set best to the record for the first student in the file
    best = makeStudent(infile.readline())
    # process subsequent lines of the file
    for line in infile:
        # turn the line into a student record
        s = makeStudent(line)
        # if this student is best so far, remember it.
        if s.gpa() > best.gpa():
            best = s
    infile.close()
    # print information about the best student
    print("The best student is:", best.getName())
    print("hours:", best.getHours())
    print("GPA:", best.gpa())
```

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