Answer Key: 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.")
```

Answer Key:

```
There are 3 fun days in a week
Two of them are Friday Sunday
My favorite day 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 $200,000 \times 0.25 + 100,000 \times 0.5 = 100,000$.

Answer Key:

```
def calculate_tax(income):
    if income < 200000:
        tax = income * .25
    else:
        tax = 200000 * .25 + (income - 200000) * .5
    return tax
```

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)
```

Answer Key:

```
def getInputs():
    fname = input('Enter file name: ')
    word = input('Enter word: ')
    return fname, word

def countWord(line, word):
    return (line.count(word))

def average(l):
    total = 0
    for i in l:
        total = total + i
    return total/len(l)

def printSummary(word, a):
    print("The average number of times per line the word", word)
    print("occurs in the file is", a)
```

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?

   Answer Key: 2

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

   Answer Key: 1024

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
```

Answer Key:

```
2
6
12
```

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

numbers.txt:

```
123456
```

Answer Key:

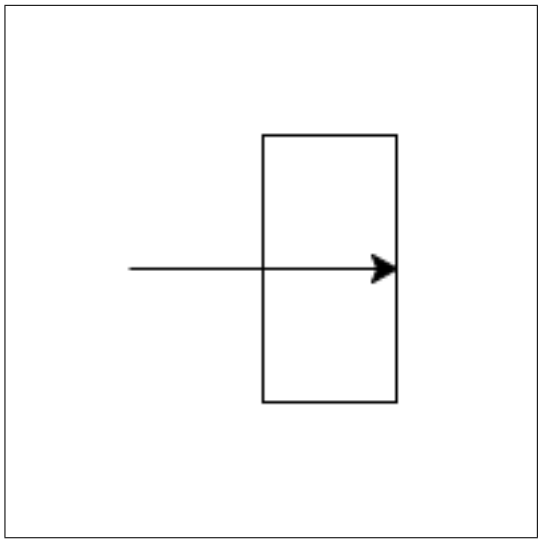
```
123456
```

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)

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


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

Graphics Displayed:

Answer Key:



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):

Answer Key:

```
def main():
    infile = open("infile.txt","r")
    for l in infile:
        if l.find("The Amazing Spider Man") != -1:
            print(l)
    infile.close()
```

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
   ```

   Answer Key:

   ```
   def find(st):
       index = 0
       location = -1
       found = False
       while not found
           if st[index] == ',':
               location = index
               found = True
           index = index + 1
       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
   ```

   Answer Key:

   ```
   def getSmaller(ls):
       for i in range(len(ls)-1):
           if ls[i] < ls[0]:
               ls[i],ls[0] = ls[0],ls[i]
   ```

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
```

Answer Key:

```
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:
                scoreB = scoreB + 1   # I added this line
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                scoreA = scoreA + 1   # I added this line
                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 >= 11 or b >= 11) and ( abs(a - b) >= 2 ) # I changed this line
```

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. **Answer Key:**

```
class Song:
    def __init__(self, name, length, artist, composer):
        self.name = name
        self.length = length
        self.artist = artist
        self.composer = composer

    def getLength(self):
```

```
        return self.length
```

(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):
```

Answer Key:

```
def tapeLength(mixTape):
    total = 0
    for song in mixTape:
        total = total + song.getLength()
    return total
```

Answer Key: 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="")
```

Answer Key:

```
March
October
January
August
MONTHS!! MONTHS!! MONTHS!!
```

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 $100,000 \times 0.25 + 100,000 \times 0.5 = 75,000$.

Answer Key:

```
def calculate_tax(income):
    if income < 100000:
        tax = income * .25
    else:
        tax = 100000 * .25 + (income - 100000) * .5
    return tax
```

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)
```

Answer Key:

```
def getInputs():
    fname = input('Enter file name: ')
    return fname

def countAs(line):
    return (line.count('A')+line.count('a'))

def average(l):
    total = 0
    for i in l:
        total = total + i
    return total/len(l)

def printSummary(a):
    print("The average number 'A' or 'a' per line")
    print("in the file is", a)
```

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?

   Answer Key: 3

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

   Answer Key: 512

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
```

Answer Key:

```
10
22
36
```

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

numbers.txt:

```
1011121314
```

Answer Key:

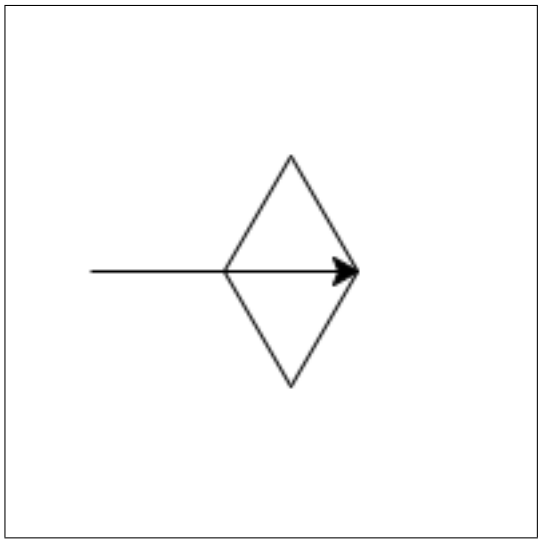
```
1011121314
```

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:

Answer Key:



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.

Answer Key:

```
def main():
    infile = open("infile.txt","r")
    lines = infile.readlines()
    numLines = len(lines)
    infile.close()
    outfile = open("infile.txt","w")
    for i in range(numLines):
        print("Awesome",file=outfile)
    print(numLines)
    outfile.close()
```

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
   ```

   Answer Key:

   ```
   def find(st):
       index = len(st) - 1
       location = -1
       found = False
       while not found
           if st[index] == ',':
               location = index
               found = True
           index = index - 1
       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
   ```

   Answer Key:

   ```
   def getBigger(ls):
       for i in range(len(ls)-1):
           if ls[i] > ls[0]:
               ls[i],ls[0] = ls[0],ls[i]
   ```

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
```

Answer Key:

```
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:
                scoreB = scoreB + 1   # I added this line
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                scoreA = scoreA + 1   # I added this line
                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 >= 25 or b >= 25) and ( abs(a - b) >= 2 ) # I changed this line
```

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.

Answer Key:

```
class Movie:
    def __init__(self, name, length, studio, director):
```

```
        self.name = name
        self.length = length
        self.studio = studio
        self.director = director

    def getLength(self):
        return self.length
```

(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):
```

Answer Key:

```
def viewLength(driveContents):
    total = 0
    for movie in driveContents:
        total = total + movie.getLength()
    return total
```

Answer Key: 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])
```

Answer Key:

```
There are 5 important subjects in school.
Don't know much about history
Don't know much about biology
Don't know much about french
Don't know much about trigonometry
But I do know that I love computer science
```

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 $50,000 \times 0.1 + 50,000 \times 0.2 = 15,000$.

Answer Key:

```
def calculate_tax(income):
    if income < 50000:
        tax = income * .10
    else:
        tax = 50000 * .10 + (income - 50000) * .20
    return tax
```

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)
```

Answer Key:

```
def getInputs():
    fname = input('Enter file name: ')
    return fname

def countSpaces(line):
    return (line.count(' '))

def minMax(l):
    return min(l),max(l)

def printSummary(m,M):
    print("The minimum number of spaces per line is", m)
    print("The maximum number of spaces per line is", M)
```

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?

   Answer Key: -2

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

   Answer Key: 128

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
```

Answer Key:

```
2
4
6
12
```

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

numbers.txt:

```
123456
```

Answer Key:

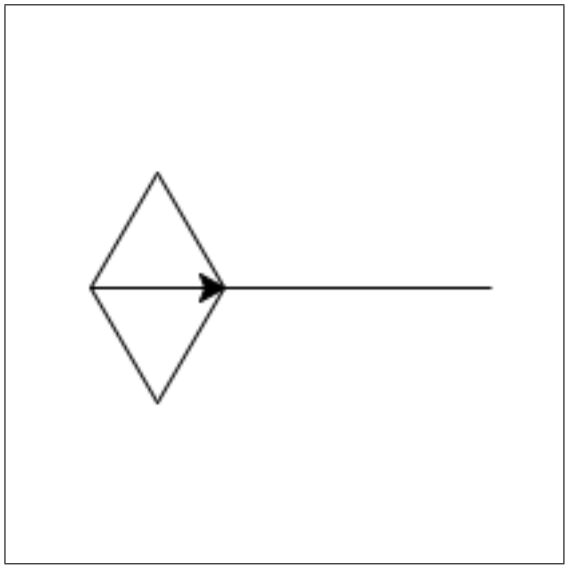
```
123456
123456
```

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)

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


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

Graphics Displayed:

Answer Key:



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

Answer Key:

```
def main():
    infile = open("infile.txt","r")
    lines = infile.readlines()
    for line in lines:
        print("-*-"+line[:-1]+"-*-")
    infile.close()
```

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
   ```

   Answer Key:

   ```
   def find(st):
       index = 0
       location = -1
       firstFound = False
       notFound = True
       while notFound and index < len(st):
           if st[index] == ',' and firstFound == False:
               firstFound = True
           elif st[index] == ',':
               location = index
               notFound = False
           index = index + 1
       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
   ```

   Answer Key:

   ```
   def getBigger(ls):
       for i in range(len(ls)-1):
           if ls[i] > ls[-1]:
               ls[i],ls[-1] = ls[-1],ls[i]
   ```

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
```

Answer Key:

```
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:
                scoreB = scoreB + 1   # I added this line
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                scoreA = scoreA + 1   # I added this line
                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 >= 21 or b >= 21) and ( abs(a - b) >= 2 ) # I changed this line
```

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.

Answer Key:

```
class Book:
    def __init__(self, title, length, author, publisher):
```

```
        self.title = title
        self.length = length
        self.author = author
        self.publisher = publisher

    def getLength(self):
        return self.length
```

(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):
```

Answer Key:

```
def libraryPages(library):
    total = 0
    for book in library:
        total = total + book.getLength()
    return total
```

Answer Key: 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] + "!!!")
```

Answer Key:

```
You have a choice of 4 options:
Omelette
Porridge
Cereal
Pancake

But I need more!!!
```

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 $500,000 \times 0.5 + 100,000 \times 0.75 = 325,000$.

Answer Key:

```
def calculate_tax(income):
    if income < 500000:
        tax = income * .50
    else:
        tax = 500000 * .50 + (income - 500000) * .75
    return tax
```

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)
```

Answer Key:

```
def getInputs():
    fname = input('Enter file name: ')
    return fname

def countSpaces(line):
    return (line.count(' '))

def calculate(l):
    total = 0
    for i in l:
        if i > 5:
            total = total + 1
    return total

def printSummary(n):
    print("The number of long lines (more than 5 spaces)")
    print("is", n)
```

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?

   Answer Key: 2

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

   Answer Key: 4

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
```

Answer Key:

```
6
8
14
```

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

numbers.txt:

```
5
6
7
8
9
```

Answer Key:

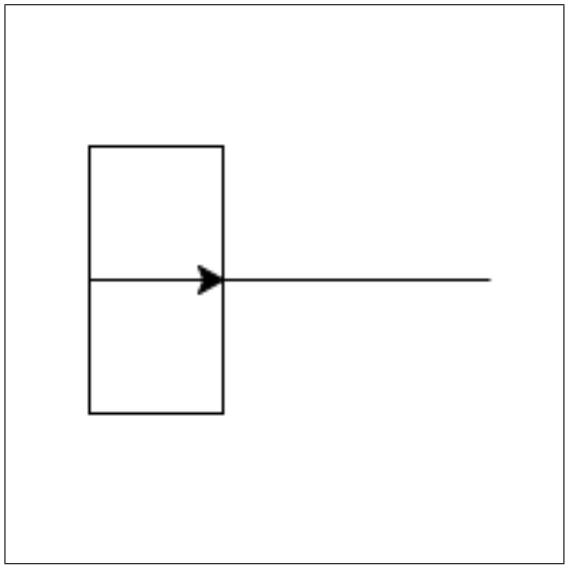
```
6
8
14
```

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.)

Graphics Displayed:

```
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)
```

Answer Key:



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".

Answer Key:

```
def main():
    infile = open("infile.txt","r")
    lines = infile.readlines()
    for line in lines:
        print(line[0].lower()+line[1:-1].upper())
    infile.close()
```

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
```

Answer Key:

```
def find(st):
    index = len(st) - 1
    location = -1
    firstFound = False
    notFound = True
    while notFound and index > -1:
        if st[index] == ',' and firstFound == False:
            firstFound = True
        elif st[index] == ',':
            location = index
            notFound = False
        index = index - 1
    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
```

Answer Key:

```
def getSmaller(ls):
    for i in range(len(ls)-1):
        if ls[i] < ls[-1]:
            ls[i],ls[-1] = ls[-1],ls[i]
```

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
```

Answer Key:

```
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:
                scoreB = scoreB + 1   # I added this line
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                scoreA = scoreA + 1   # I added this line
                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 >= 11 or b >= 11) and ( abs(a - b) >= 2 ) # I changed this line
```

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.

Answer Key:

```
class Athlete:
    def __init__(self, name, numberOfMedals, country, sport):
        self.name = name
        self.numberOfMedals = numberOfMedals
        self.country = country
        self.sport = sport

    def getNumberOfMedals(self):
        return self.numberOfMedals
```

(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):
```

Answer Key:

```
def overallMedalCount(team):
    total = 0
    for athlete in team:
        total = total + athlete.getNumberOfMedals()
    return total
```