Intro. to Computing

Name:_____

1. Turtle Graphics is discussed in sections 7.1 and 7.2 of the textbook and is NOT a standard module of Python. The objectives for looking at the Turtle graphics module are:

- Gain experience using object-based programming using existing classes, objects, and methods (p. 250)
- Understand simple graphics operations to draw 2D shapes

The Turtle methods from the Turtle Graphics module are:

Turtle Method	Description
`t = Turtle()`	Creates a Turtle object and opens a 200 x 200 pixel drawing area.
`t = Turtle(width, height)`	Creates a Turtle object and opens drawing area with the specified size.
`t.getWidth()`	Returns the width of t's drawing window.
`t.getHeight()`	Returns the height of t's drawing window.
`t.home()`	Move t to center pointing north without drawing any lines.
`t.setDirection(degrees)`	Points t in specified direction (0 is east, 90 is north, 180 is west, and 270 is south)
`t.turn(degrees)`	Adds the degrees to t's current direction (+ clockwise, neg. counter-clockwise)
`t.down()`	Puts t's tail/pen down
`t.up()`	Puts t's tail/pen up
`t.move(distance)`	Moves t forward the specified distance.
`t.move(x,y)`	Move t to the specified x-y coordinate
`t.setColor(r, g, b)`	Sets the color of the pen to specified RGB value
`t.setWidth(penWidth)`	Sets the width of the pen in pixels (default is 2)

The following program (Figure 7.4) causes the turtle to move 30 times for a distance of 20 pixels each in a random direction with each move drawn in a random color.

```
"""
File: randomwalk_randomcolors.py
Description: A turtle takes a random walk drawing a random color.
"""

from turtlegraphics import Turtle
import random

def randomWalk(turtle, turns, distance = 20):
    turtle.setWidth(1)
    for x in xrange(turns):
        turtle.turn(random.randint(0, 360))
        turtle.setColor(random.randint(0,255),  # red component
                        random.randint(0,255),  # blue component
                        random.randint(0,255))  # green component
        turtle.move(distance)

randomWalk(Turtle(), 30)
temp = raw_input("Hit the <Enter> key to quit")
```

a) Define the following functions:

- drawLine - this function expects a Turtle object and four integers as arguments. The integers represent the end-points of a line segment. The function should draw a black line segment of width 1 with the turtle and do no other drawing.

Write a main function that uses drawLine to:

- create a Turtle with a window size of 400 by 400, and
- draws a line from (150, 150) to (-200, -200).

2. Table 7.4 contains the `Image` object methods from the `images` module:

Image Method	Description
`i = Image(filename)`	Loads and returns an image from filenam. Raises an error if the filename is not found or the file is not a GIF file.
`i = Image(width, height)`	Creates and returns a blank image with the given dimensions. The color of each pixel is white and the filename is the empty string.
`i.getWidth()`	Returns the width of `i` in pixels.
`i.getHeight()`	Returns the height of `i` in pixels.
`i.getPixel(x, y)`	Returns the tuple of integers representing the RGB values of the pixel at position `(x,y)`
`i.setPixel(x, y, (r,g,b))`	Replaces the RGB value at position `(x,y)` with the the RGB value given by `(r,g,b)`
`i.draw()`	Displays `i` in a window. **The user must close the window to return control to the method's caller.**
`i.clone()`	Returns a copy of `i`.
`i.save()`	Saves `i` under its current filename. If `i` does not yet have a filename, `save` does nothing.
`i.save(filename)`	Saves `i` under filename. Automatically adds a `.gif` extension if filename does not contain it.

The textbook example below copies and blurs a colored image. Notice the nested for-loops that processing each pixel.

```
def blur(image):
  """Builds and returns a new image which is a blurred copy of the
     argument image."""

  def tripleSum((r1, g1, b1), (r2, g2, b2)):
     return (r1 + r2, g1 + g2, b1 + b2)

  new = image.clone()

  for y in xrange(1,image.getHeight()-1):
     for x in xrange(1, image.getWidth()-1):
         oldP = image.getPixel(x, y)
         left = image.getPixel(x-1, y)
         right = image.getPixel(x+1, y)
         top = image.getPixel(x, y-1)
         bottom = image.getPixel(x, y+1)
         sums = reduce(tripleSum, [oldP, left, right, top, bottom])
         averages = tuple(map(lambda x: x / 5, sums))
         new.setPixel(x, y, averages)

  return new
```

a) What does the statement: `sums = reduce(tripleSum,[oldP,left,right,top,bottom])` do?

b) What does the statement:
`averages = tuple(map(lambda x: x / 5, sums))` do?

3. Write a function to enlarge an image by a specified factor.