Final Examination

Comp 123
Fall 2016
Daniel Kluver

Notes:

- This test is to be taken on the computer.
- Normal testing rules apply - You may use any digital resource reachable from Moodle, any digital file you have created, any notes you have created, and the paper textbook.
- **Do Not Hesitate to Ask Questions** if something is unclear, or if you can't remember something, and I'll give you a hint.
- By now you should have downloaded the `FinalFiles.zip` archive and extracted it to a folder. Copy the contents of this folder into a new project in pycharm. This folder has all resources you should need to complete the final.
- This exam has each question split into its own file (Q1.py through Q6.py) Please answer each question in its own file and make sure that your submission includes all files.
- Each question file has a few lines of comments at the top which are there to remind you of your task on this question. These are not meant to replace this PDF.
- When you are finished with the exam, create a new zip archive of the folder containing all your files and upload it to Moodle. **Double-check that you are handing in the right files!**

1. (20 pts) In `Q1.py` you should find a recursive function - `palindrome`. This function takes one input - a string and returns True if the string is a palindrome and False if the string is not a palindrome. A Palindrome is any string that is the same forwards and backwards for example, "a". "abba", "applppa" and "qwerrewq" are all palindromes, "asdf", "ape", and "monkies!" are not palindromes. More examples of what is and is not a palindrome are available in the Q1 file.

   In this question you have several tasks:

   (a) Using comments, label each line of code as either part of a base case or a recursive case. If there are more than one base case or recursive case number the cases (I.E. Recursive case 1)

   (b) In comments, after the function explain each case. For base cases, please explain what the case is (under what conditions that code will get ran) and why the base case is correct (Under the condition it is ran why can we be sure of the answer). For recursive cases please explain how the recursive call moves closer to a base case.

2. (20 pts) In `Q2.py` write a function `greater(aDict)` which takes one parameter - a dictionary which has numbers for both keys and values. Your job is to return a list that contains every key in the dictionary that is larger than its value. The original dictionary should not be modified An example might make this more clear.

```
aDict = {1:2, 2:1, 306:306, 4:1, 51:61, 17:3}
print(greater(aDict))
#should print (order may change)
# [2,4,17]
```

The list contains the values 2, 4, and 17 as those are the only keys in the dictionary that are strictly larger than their value. The lists returned may have any order, so long as the values are correct.

3. (20 pts) In `Q3.py` write a function named `brightestImage(image1, image2)`:. This function will take two parameters, both are images. The function should return a new image make by mixing these two images. For each position in the new image the color should be from one of the two images based on which image is brighter at that location. Brightness should be based on luminance - the average of the r, g, and b values for that pixel. In this way the image will have some pixels with colors from the first image, and some pixels with colors from the other image.

   This process results in some pretty cool (in my opinion) looking images. Note, that the images are not guaranteed to be the same size, you should create your new image to have width equal to the minimum of the two source image widths and height equal to the minimum of the two source image heights. Below is an example of what to expect:

image1 (30% scale for pdf)

image2 (30% scale for pdf)

(see next page for output)

Resulting picture, (image at 70% scale for pdf)

Note, this image may look like colors are being *blended*, but they are not, each pixel is either entirely motorcycle or entirely cat. Since the cat image and the water are both almost equally dark in places there are parts of the image where cat and swan pixels are equally likely. Other parts of the image (the pumpkin toy and the swan) are both very clearly entirely swan or pumpkin.

4. (20 pts) In `Q4.py` you will find code for a tkinter interface. In this question you will complete this tkinter interface. The interface is designed to help students understand even and odd numbers. The interface has one entry, one button, and one label. The user enters a number into the entry, and then presses the button. The number that is entered is copied into the label, and the label's background changes to green if the number is even, or red if the number is odd.

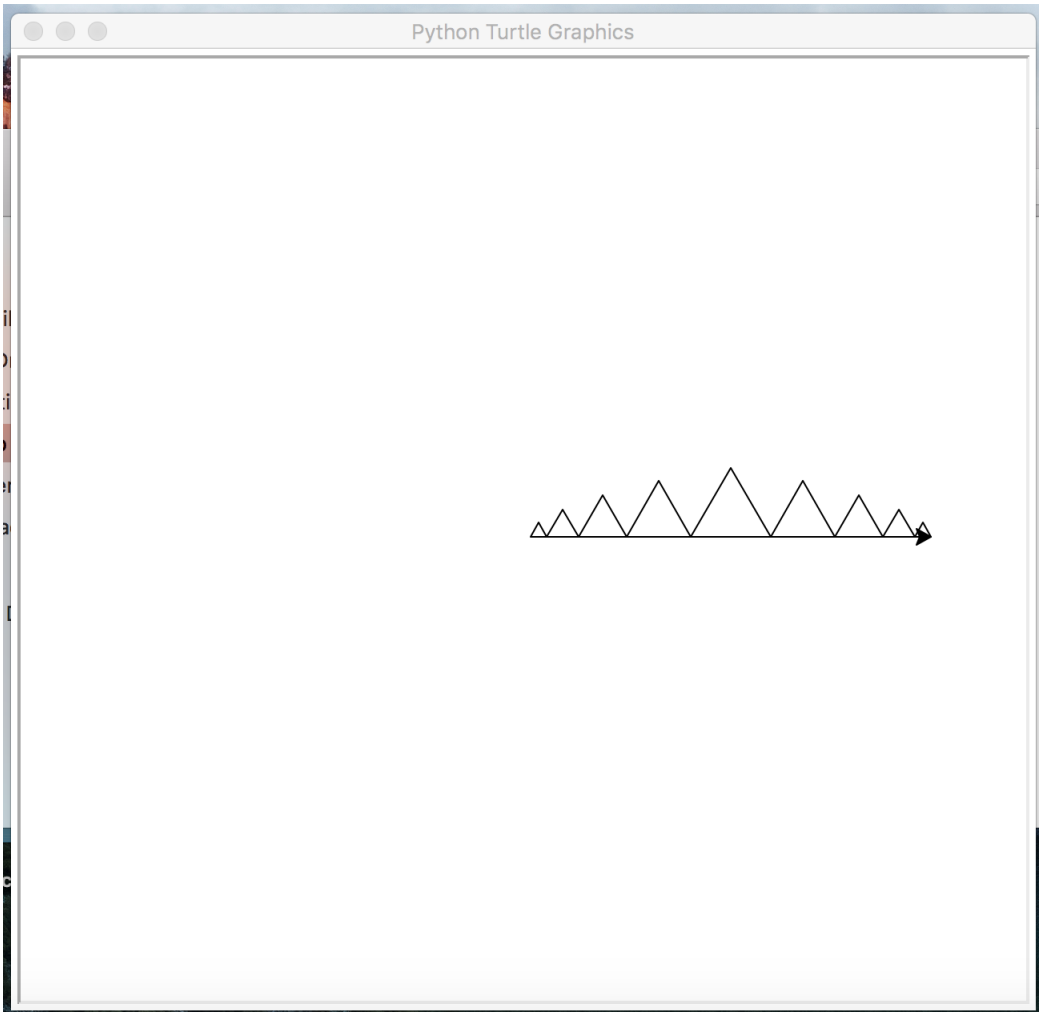
   You are free to assume that the user will only type integers into the entry. That is, you do not have to check if the text entered can safely be cast to a number and it is OK if your code fails when a non-number is typed.

   Currently, only the widgets have been placed (to save you time placing widgets). Your job is to complete this interface so that the button performs the operation described above. To do this you may find it useful to define extra variables, and you may want to change the provided code to have commands and string variables (the current code is not written to use string variables so if you want to use them you will have to add them).

   Note: this interface has several labels that are there to help instruct the user. Mostly, you will want to ignore them and focus on the main entry, button, and label.

5. (20 pts) In `Q5` write a turtle function `drawTrianges(turt, max)`. When called, this function should make the turtle draw a series of equilateral triangles, each right next to the other. The triangles should start small (10 pixels to a side) and get bigger (by 10 pixels) each triangle until they reach a maximum size, after which they should get smaller (by 10 pixels) until they reach 10 pixels again.

The first input to the function should be a turtle which will be used to draw the triangles, the second input should be the maximum size that the triangles will reach. This function does not need to return anything. An example of calling this function with max size 50 is presented below:



The result of calling `drawTriangles(turt, 50)`. Note image is at 60% size for pdf.

6. (20 pts) In `Q6.py` there is a function `wordCount(fileName)`. This function takes the name of a file (a string) as its only parameter. The function reads through the file line by line splitting the line at white space and counting the words on that line (which is accumulated to count the words in the file). You could imagine using the function as a basic tool to help you hit minimum word counts for a function.

   This function has two bugs in it. Your job is to find, describe, and fix these bugs. For each of the two bugs:

   (a) Use a comment to mark where the bug is, and describe what is wrong with the code.

   (b) Fix the bug

   (c) In a comment, describe how you fixed the bug.