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 `def rev(aDict)` which takes one parameter - a dictionary. Your job is to "Reverse" this dictionary. What this means is that you will take a dictionary which maps keys to values and return a new dictionary where the old values are the new keys and the old keys are the new values. An example might make this more clear.

```
someDict = {1:'a', 2:'bee', 3:'sea', 'alpha':1, 'beta':2, 4:'b'}
print(rev(someDict))
# should print the following (order may change)
# {'a':1, 'bee':2, 'sea':3, 1:'alpha', 2:'beta', 'b':4}
```

As you can see from the above example, if the input dictionary has a key-value pair `4:'b'` then the returned dictionary should have a key value pair `'b':4` Please do this using a for loop.

3. (20 pts) In `Q3.py` write a function named `mergeRGB(rImage, gImage, bImage)`. This function will take three parameters, each one is an image. The function should return a new image make by merging each image. The red values for the new image should be the red values from the first parameter `rImage`. The green values for the new image should be the green values from the second parameter `gImage`. The blue values for the new image should be the green values from the third parameter `bImage`.

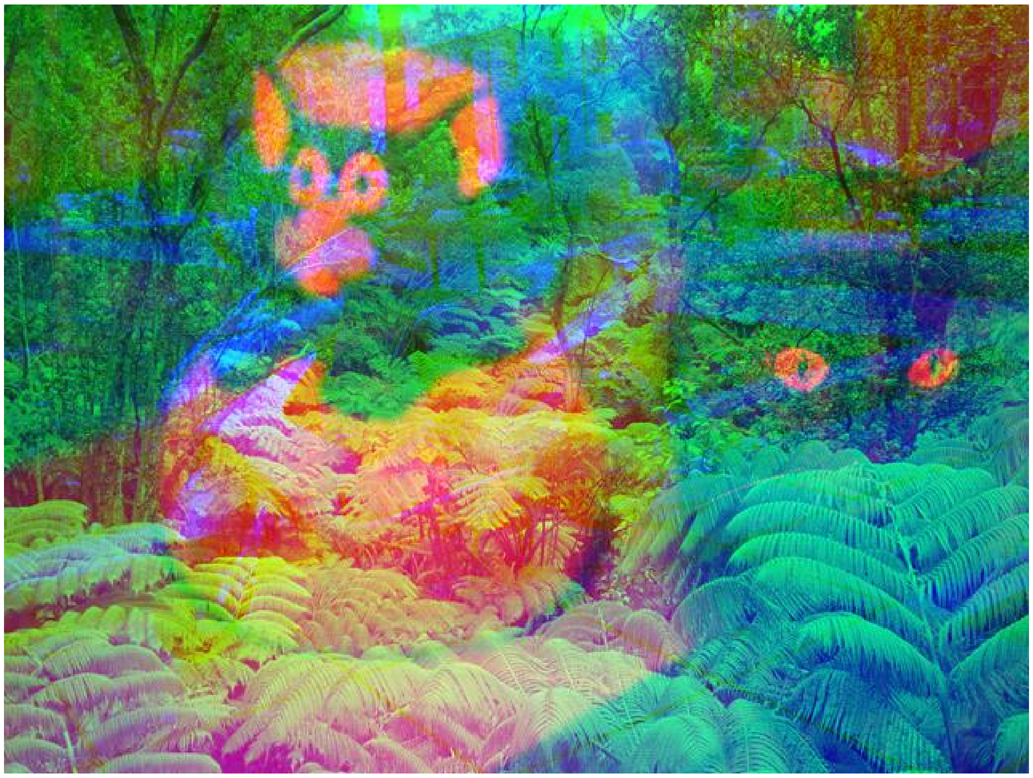
   This process results in some pretty cool (in my opinion) looking images. Note, that the images are not guaranteed to be the same size, you should create your new image to have width equal to the minimum of the three source image widths and height equal to the minimum of the three source image heights.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

rImage (20% scale for pdf)

gImage (20% scale for pdf)

bImage (20% scale for pdf)

(see next page for output)



Resulting picture, it is easy to see the red and green images, it is a little harder to see the blue motorcycle image, but it is there. (image at 60% scale for pdf)

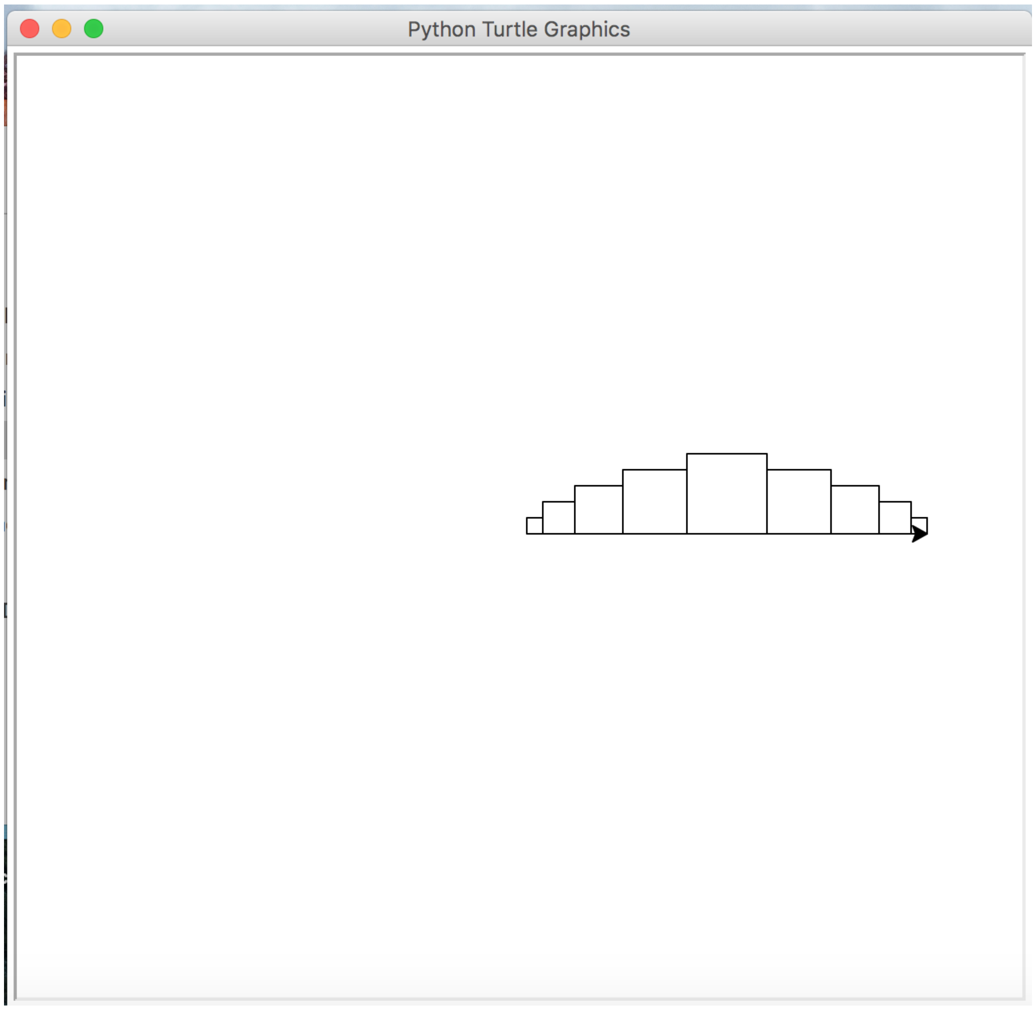
4. (20 pts) In `Q4.py` you will find code for a tkinter interface. In this question you will complete this tkinter interface. The interface is designed to help students understand how the python `and` works. The python and (also known as boolean logic and) is an operator for boolean logic that takes two inputs and is true if and only if both inputs are true. If either input is false the and of those inputs is false. The interface shows the situation of `x and y` which will be `True` if and only if `x` and `y` are both true. The interface has two buttons which the user can use to toggle the value of `x` or `y`. If the button is pressed leading to `x` and `y` equaling `True` then the interface should also show that `x and y` is true.

   Currently, only the widgets have been placed (to save you time placing widgets). Your job is to complete this interface so that the two buttons toggle the `x` and `y` values and if both are true also updates the result label. 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 three labels that are used to show `x`, `y` and `x and y` respectively.

5. (20 pts) In `Q5` write a turtle function `drawSquares(turt, max)`. When called, this function should make the turtle draw a series of squares, each right next to the other. The squares should start small (10 pixels to a side) and get bigger (by 10 pixels) each square 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 squares, the second input should be

the maximum size that the squares 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 `drawSquares(turt, 50)`. Note image is at 60% size for pdf.

6. (20 pts) In `Q6.py` there is a function `def grep(fileName, searchWord)`. This function takes the name of a file (a string) and another string (the search word). The function opens the file and searches through it line by line looking for the search word, printing each line of the file which contains the search word. You could imagine this function being used as a primitive search tool.

   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.