CS 1120: Media Computation Spring 2017
Assignment 3: Making a collage: a more general approach

Due: Monday, February 27, 2017, by 11:59 p.m.

This assignment is worth **50 points**, which accounts for **4.375%** of your final grade

For your last assignment, you generated a collage using 4 posterized images as your sources. Instead of writing the same code 4 times, you may have used functions and kept your code concise and free of duplication. However, although that approach was a strong recommendation, it was not a requirement.

The goal of this assignment is to place you in a situation where you are forced to call a function - not only because writing out the code would make your program too long, but also because you don't know how many times you would need to perform a certain action.

Specifics

Write a program that takes a filename and an integer n as input and produces a collage of size n-by-n.

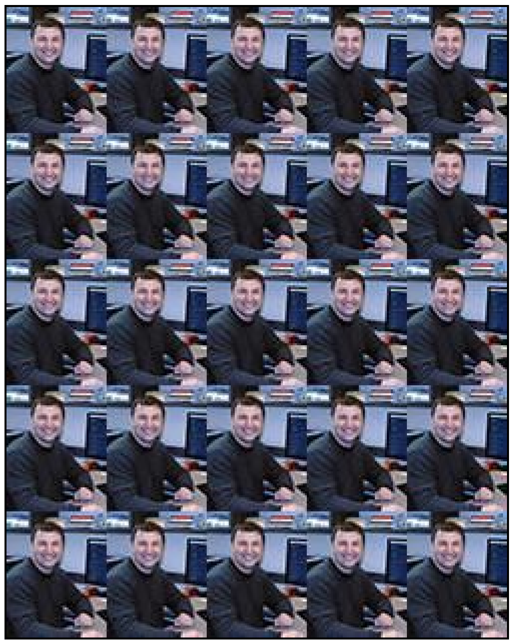
Your program should consist of at least 2 functions:

makeCollage(filename, n) #returns the collage picture object
copy(source, target, x, y) #doesn't return anything

The makeCollage function is the main function - that's the function you want to call to execute your program. The **copy** function is called from the **makeCollage** function for each instance of your source picture being copied onto the new "canvas".

HINT: use a very small image (50-100 pixels in either dimension), otherwise your program may take very long to execute (remember that JES was not built for speed).

For example, **makeCollage('mini-sergey.jpg', 5)** will produce this:



MORE HINTS:

- The easiest way to make a n-by-n collage is to use a nested for loop
- makeCollage is a relatively simple function:
    - it creates a blank picture object of an appropriate size, and
    - passes it several times to the copy function together with the source picture object
- the copy function needs to know what it should copy, onto what target "canvas" it needs to copy it (i.e., where does it paste it?), and where exactly on the canvas is the copy placed
- the copy function is very similar to what we did in our last lab

Submit your work

Submit 3 files to eLearning:

1. Your file **assignment3.py**.

2. The image you used to test your code

3. Your generated collage as a jpg file

Extra Credit

This will earn you up to 10 extra points, as well as bragging rights. However, you may earn extra credit only if you have completed the main part of the assignment (i.e., you need to submit this code in addition to your standard makeCollage implementation.

If you look at your copy function, you'll see that it's pretty boring: it gets a color from a pixel, and it sets that color to a pixel on the canvas. But what if it first transformed that color?

Create an alternative function, copyTransformed. It does the same thing as copy, except for each pixel it calls a **getTransformedColor** function. That function should take the current pixel as an argument (and maybe something else), and return a color generated based on the current pixel's color. You know how to create a posterized effect, convert to grayscale, negate, etc. Choose any effect. That way you will be not just copying the same pixel, you will be making it different.

Make sure you do not include the color transformation code in the copyTransformed function - use a separate function for that.

The above approach will earn you up to 5 extra points (more complex effects = more points). But if you want the full 10 extra points - read on.

Of course, you don't have to apply the same effect to each copy (although you need to apply the same effect to each pixel in a given copy. One thing you could do is modify the source color's red, green and blue channels by some factor. You could even generate 3 random factors each time the copyTransformed function is called, and use those factors to modify the original's red, green and blue values. To do that, you would need to do 2 things:

1. At the top of your program include this line:

import random

This statement will make Python's random module available to your code.

2. To get a random factor between 0 and 2, use this method:

factor = random.uniform(0, 2)

This will give you a factor (as a float data type) that you can use to reduce or increase your color's intensity(you may ignore the possibility of going over 255 - JES will take care of that for you). For example, 0.42 will reduce the intencity, 1.42 will increase it.

Here's what I generated:

Submit your work

The same submission requirements apply.