MA 151: Project #3

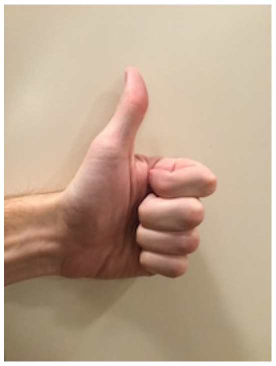
due Friday December 15 (final exam day)

Submissions

This project will be completed in groups of size 1-3, and submitted in the usual way on our class website. One member of each group should submit the code, and all group members' names should be listed in comments at the beginning of the file.

Overview

This is the last episode of a 3-part project which will focus on digital images. This time we will use actual "real world" image files which look something like:



There are many ways that people store digital images in the real world. For photographs, the most common file type these days is JPEG. These files usually have filenames ending in ".jpg".

From our class webpage you can download `project3package.hs`, which is a module that includes some functions you'll want to use. Put that file in the same directory as your code file, and put `import Project3package` at the top of your code file.

This module includes the data declaration for the `Pixel` and `Image` types, and also two functions:

```
readJPG  :: FilePath -> IO Image
writeJPG :: FilePath -> IO ()
```

These two functions work just like `readFile` and `writeFile`. I created them using the functions from the `JuicyPixels` package, so you need to have that installed in order for this to work.

Required functions

You will want to paste in most of the functions from Project 2, though you won't need them all. The data declarations are part of `project3package.hs`, so you shouldn't include them.

Some more image functions

- A function called `randomImage` which works just like `blackImage` and `whiteImage` from Project 1, but makes an image of the given size where each pixel individually has a random color.
- A function called `darken` which takes an `Int` and darkens the image by the given amount. You can darken a pixel by adding some number into each of its three color channels. Make it so that if I had a pixel like `RGB 100 50 200` and did `darken 80`, the resulting pixel would be `RGB 180 130 255`. (Make sure that the numbers never exceed 255.) Make sure the function correctly handles negative numbers, which will brighten the picture.
- A function called `colorSwap` which swaps the red, green, and blue channels. Make it so that if I had a pixel like `RGB 100 50 200`, then after the color swap that pixel would be `RGB 50 200 100`. The resulting image will look the same but with the colors all weird.
- A function called `noisy` which takes an image and color swaps some of the pixels at random (the color swapping should be just like above). Make it so that each pixel has a 10 percent chance of being swapped. The resulting image should look mostly the same but with little specks all over it.

View

- A function called `view :: Image -> IO()`, which takes an image, writes it into a file in the current directory, and then opens it on your computer so you can see it. (Use `callCommand` like we did in class. In order to open the image on your computer, you'll first need to write it out into a file.)

  You can test the `view` function in GHCi by doing something like:

  ```
  i <- readJPG "thumbphoto.jpg"
  ```

  ```
  view i
  ```

- Every time you use `view` to look at an image, you will end up creating a jpg file on your computer. Write a function called `cleanup` which deletes any files that were created by `view`. (Be very careful when testing this- make sure you don't delete your code file.)

An interaction

Write a function called `pictureTime` of type `IO ()` which collects your functions into an interaction. The interaction should begin by asking the user for the name of a file to load. Then there should be some kind of menu of options to choose from. The options should include:

- view the image
- open a new file
- save the current image to a file (ask the user for the name)
- quit the interaction
- at least 6 choices to manipulate the image in some way (add a border, rotate it, lighten or darken, etc). At least 2 of these choices should be ones which prompt the user for more input. Like darken will need to ask the user to say how dark to make it.

Your interaction doesn't have to behave exactly like mine does. But here are some features that I want to see:

- the interaction should "remember" the manipulations from one step to the next. So if I rotate the image, then darken, and then view, it should be rotated and darkened.

- after each user command, there should be some kind of "OK I did it" message, and then the menu should display again.
- the user should never be able to trigger the `No such file or directory` GHCi error. This means that you'll always need to make sure the file exists before you try to open it.
- the user should also never be able to trigger a nonexhaustive patterns error.
- after every user command, and before the user quits, you should call the `cleanup` function so that no temporary files will accumulate.

Extra credit!

If you want, you can add in some extra features to earn up to 5 points of extra credit. (So your maximum score will be 25 out of 20.) Here are some suggestions:

- (3 points) Make a function that superimposes two images of the same size by averaging their colors pixel by pixel. For 4 points, make it so that works even when the sizes aren't the same by cropping out any "overhanging" parts. For example when you superimpose the thumb with its rotation, you get the first picture below.
- (5 points) Make a function which takes an integer n and an image, and chops up the image vertically into n pieces, randomly rearranges them, and then puts the image back together. For example if you do this kind of scrambling into 10 pieces, you get the second picture below.
- (3 points) Make functions called `doubleScale` and `halfScale` which either magnify the image or shrink the image. For 4 points, make functions called `scaleUp` and `scaleDown` which take an Int parameter and scale the image by that factor.
- (2 points) Make a function called `vGradient` which progressively darkens the image vertically. The top of the image should be its original brightness, but the bottom should be all black. (So every row is slightly darker as you go down.) See the third picture below.
- (? points) Whatever else you can think of! Tell me your idea and I'll tell you how many points it'll get you.