CS 101
Program 9/10
Object-oriented Image Editor

Program 9 & 10 algorithm due Sunday night, Nov 18.
Program 9 (class implementation) due Sunday night, Dec 2.
Program 10 due Sunday night, Dec 9.

For the final project this semester, we'll use two assignments to build up a single large program. Program assignment 9 is to write a PPMImage class, including the functionality from program 5 plus some additional features. Program 10 will be the menu-driven program that uses the PPMImage class to provide a user-friendly image editor.

The pixel and PPMImage classes
Program assignment 9 consists of 2 classes: a pixel class, and a PPMImage class.

The pixel class:
This class is very simple. It represents a single pixel. It has 3 data elements: self.Red, self.Green, and self.Blue. It also has the following methods:

- __init__(self, Vals = (0,0,0)): The initializer takes 3 values for the initial R, G, B values. If any of them are non integers, this method should raise a TypeError. If any of them are less than 0 or greater than 255, this method should raise a ValueError. (Note that raising an error does NOT require a try/catch block.)
- __str__(self): This should return the pixel's three color values, in order R, G, B, in a string with no leading or trailing space, and exactly one space character between them. If the pixel has values, say, R= 100, G = 75, and B = 200, this method would return the string:
  `'100 75 200'`
- __repr__(self): This method returns the RGB values, in that order, *as a tuple of integers*. In the above example, this method return the tuple `(100, 75, 200).`

The pixel class should be in its own module class (pixel.py).

The PPMImage class:
This class represents the image itself. It contains several data values:

- Name: A string, the name of the file containing the image. This defaults to the empty string.
- Rows, Cols: The number of rows of pixels, and the number of pixels per row. These both default to 0.
- Image: A list of lists. Each internal list is a row of pixels; Image is a list of rows. This defaults to the empty list.
- UnsavedChanges: A boolean variable that tracks whether the image has been altered since the last time it was saved. This defaults to False.

The PPMImage class has the following methods:

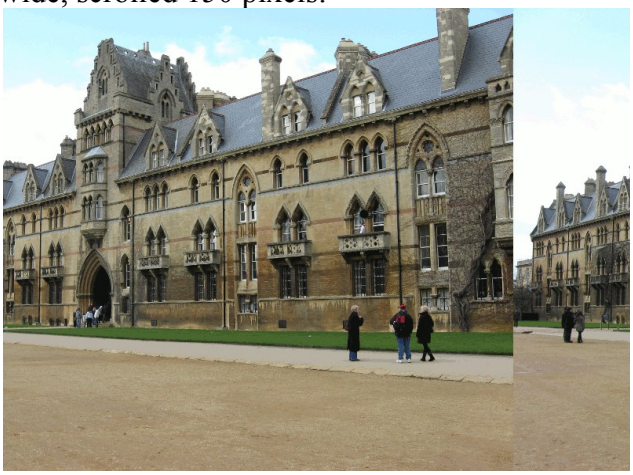
- __init__(self, Filename=''): The initializer. It sets all default values as specified above. Then, if a filename was specified, calls `self.Open(self.Name)`.
- __str__(self), __repr__(self): These both return the same thing: A string consisting of "PPMImage " followed by the name of the file, a comma and space, then the number of rows and columns:
  `PPMImage tokyo.ppm, 600x800`

- Open(self, Name): Attempts to open the specified file and read the image data. If opening the file fails, Open() should print “Unable to open file” and return. If the first thing in the file is anything other than the text constant “P3”, the program should print “This is not a PPM file.”, close the file, and return. Otherwise, Open() should read the image data (rows, colums, maximum value, then the pixel data) and fill the internal data structures. Remember that self.Image is a list of rows, and each row is a list of pixels, *not* a list of integer tuples as program 6. When finished, Open() should close the file and set self.UnsavedChanges to False.
- Save(self): This method saves the file using the current value of self.Name as the file name. If the file can't be opened (an IOError occurs), Save() should print a message including the file name: “Can't open tokyo.ppm for saving.”, for example, and return. Otherwise it should save the file in the standard PPM format. (Note that the pixel.__str__() method will come in handy here.) When finished, this method sets self.UnsavedChanges to False and closes the file.
- SaveAs(self, NewFileName): This method saves the file under a new name. If the file can't be opened, it prints an error message and returns, the same as Save(). Otherwise self.Name is set to the new file name, the file is saved under the new name, and self.UnsavedChanges is set to False.
- Grayscale(self): This sets the pixels in self.Image to the necessary values to make the image grayscale, and sets self.UnsavedChanges to True. If you base your code on the Program 6 solution, give credit for it in the comments.
- Negative(self): This sets the pixesl in self.Image to the necessary values to make the image a negative, and sets self.UnsavedChanges to True. If you base your code on the Program 6 soludtion, give credit for it in the comments.
- Lighten(self): This alters all pixel values by increasing all pixel values by 50, but not exceeding Max (255). It also sets self.UnsavedChanges to True.
- Darken(self): This alters all pixel values by decreasing all pixel values by 50, subject to a minimum value of 0. It also sets self.UnsavedChanges to True.
- CountColors(self): This function returns an integer, the number of distinct pixel values present in the image. Two pixels have distinct values if any of their components differ: different Red values, different Green values, or different Blue values. This does not modify any of the object's data.
- ReduceColorspace(self): The PPM and JPG formats can support thousands of distinct colors in a single image. Other formats,, such as GIF, cannot. Thus, we may have to reduce the number of colors present in the image before changing formats. This method transforms the image data as follows: Any pixel value <= 50 gets the new value 25; any pixel value in the range 51 <= x <= 100 gets the new value 75; any pixel value in the range 101 <= x <= 150 gets the new value 125; a value in the range 151 <= x <= 200 gets 175; and any other value gets the new value 225. This is done independently for the Red, Green, and Blue components of each pixel. This also sets self.UnsavedChanges to True.
- ***Extra credit!*** For 10 points extra credit, add the method Scroll(self, NCols): This method 'scrolls' the image left to right a specified number of columns, wrapping pixels around from the left side back onto the right. Obviously, this sets self.UnsavedChanges to True as well.

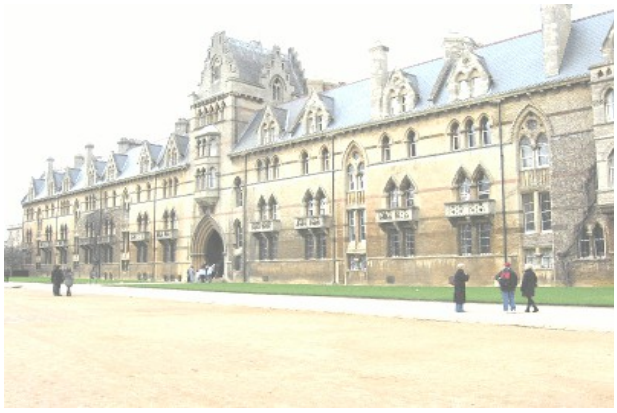
Program 10:
The last programming assignment of the year is a menu-driven program that presents a user interface to the PPMImage class. It should offer the user the chance to open a file, save the file, save under a new name (Save As...), apply any of the image filters, or to quit. If the user says to quit or to open a new file, the program should check if the image has been changed since it was last saved, and if so, ask the user whether to save the changed image or to abandon changes. The program should use functional decomposition as appropriate; if carrying out an operation will require more than a couple of lines of code, it should probably have its own function. The program should loop until the user says to quit.

Sample Images:

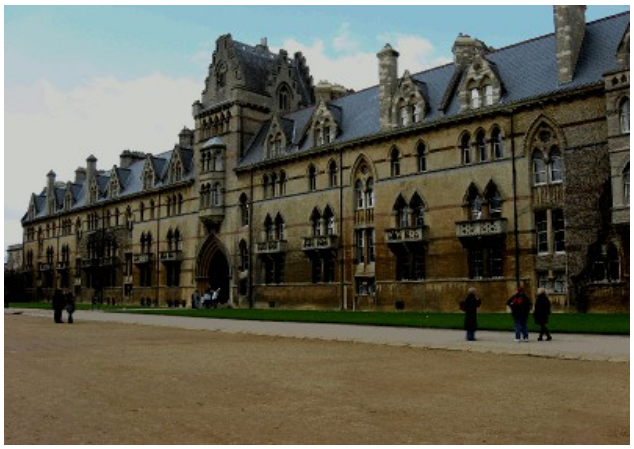
Hogwarts.ppm, 800 pixels wide, scrolled 150 pixels:



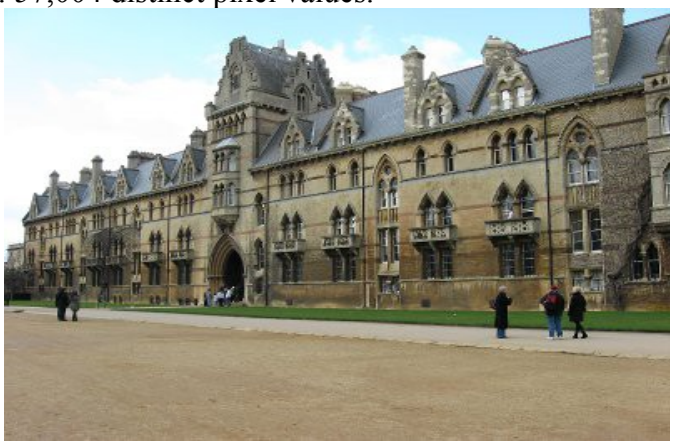
Hogwarts, lightened 2 steps:



Hogwarts, darkened 1 step:



Original Hogwarts image: 57,004 distinct pixel values.



Reduced-colorspace image (42 distinct pixel values). Note that some colors are distorted. Our method is not a good general method, but works well enough for this example: