Image Puzzles
Project 1.0 – CS 2308 – Spring 2016

Due: Wednesday, 1/27/16 @ noon
35 points

PROBLEM STATEMENT

On TRACS under the ‘Resources’ tab for this course, in the ‘Assignment Support’ folder, you’ll find a file Project1A_Files.zip to download. Inside are 3 images: puzzle1.ppm puzzle2.ppm puzzle3.ppm

These images are in PPM (Portable Pix Map) format, an image format that encodes images in ASCII text. If you’re interested, you can find more information about this image format here: https://en.wikipedia.org/wiki/Netpbm_format (however, understanding all the details of the PPM format is not necessary to complete the assignment).

PPM images represent a picture as a 2-D grid of RGB (Red-Green-Blue) values. Each pixel of the image is a tiny square whose color is defined by a triplet of values representing how much red, green, and blue are present in the pixel. For example, a black pixel is represented by a value of 0 0 0, and a white pixel by 255 255 255.

The three puzzle images on TRACS are all pictures of famous objects. However, they have all been distorted beyond recognition. Your job is to undo the distortions, reveal the true images, and identify the famous objects.

In the same zip file you’ll find a C++ file, imagePuzzles.cpp. This program contains code to read in PPM images from the same directory where the code is run, a main function that calls three empty recoverImageX() functions (one for each image) to un-distort the pictures, and code to write the solution PPM images to disk.

You only need to add code to the recoverImageX() function bodies.

Do not re-name the puzzleX.ppm files or imagePuzzles.cpp will not work. Images created by your recoverImageX() functions will be written to disk (in whatever directory you run your program in) as solution1.ppm, solution2.ppm, and solution3.ppm. You can then open these images to view the solutions created by your functions.
Each RGB pixel in the image is encoded as a structure, `Pixel`, defined as follows:
```c
struct Pixel {
    int red;
    int green;
    int blue;
};
```
The images are represented as 2-dimensional arrays of `Pixel` structures, with 300 rows and 400 columns.

Below is a description of the distortion that has been applied to each image. Your code in `recoverImage1()`, `recoverImage2()`, and `recoverImage3()` should attempt to undo the described distortion.

**puzzle1.ppm**
The blue and green values are all just meaningless noise (random values) added to obscure the real image, which is entirely encoded in the red pixels. The red values have all been divided by 10, so they must be scaled back up by a factor of 10. Throw away the meaningless blue/green values (i.e., set them to zero).

**puzzle2.ppm**
Here, the true image is in the blue and green values. The red values are all meaningless noise that have been added to obscure things: get rid of them. The blue and green values have all been divided by 20, so they need to be scaled back up by a factor of 20.

**puzzle3.ppm**
The true image is all in the blue values; the red and green values are meaningless noise added on top to obscure the true image. The hidden image is encoded using only the blue values that are less than 16 (i.e., 0 through 15). If a blue value is less than 16, multiply it by 16 to scale it back up to its proper value. If a blue value in the original image is already 16 or more, it is random garbage and should be removed (i.e., set to 0). This should yield the original image, entirely in the blue channel.

However, to look more correct, the image should be all in the red channel. As a final fix, change your code to also move the values from the blue channel to the red channel.
ASSIGNMENT SPECIFICS

This assignment is intended to be a (hopefully slightly more exciting) “Hello world!” program, in that I’ve given you the bulk of the solution already and there’s very little coding left to do.

Use this assignment to re-familiarize yourself with CS 1428 concepts like C++ program structure, arrays, structs, conditionals, and loops, and to gain some familiarity editing, compiling, and running programs in a Linux environment.

The lab machines in DERR 231 include image viewers that can display PPM images. I therefore recommend that you work on this assignment in the CS Linux lab.

If you would like to be able to view PPM images on your home computer, you’ll need to download GIMP (http://www.gimp.org/) or some other open-source image program that supports PPM. On Macs, ToyViewer is a good lightweight PPM viewer (https://itunes.apple.com/us/app/toyviewer/id414298354. On Windows, IrfanView is a small photo editor that handles PPMs (http://www.irfanview.com/).

All your code should be in the function bodies for the recoverImageX() functions. You should read and understand the rest of the code, but you do not need to modify any other function.

Make sure your executable is run in the same directory as the puzzle.ppm images.

Please follow the course style guidelines, which you can find here: http://cs.txstate.edu/~mo1162/cs2308/docs/StyleGuide.pdf

imagePuzzle.cpp provides a good example of the amount and style of commenting I expect to see in your code later on.

SUBMISSION INSTRUCTIONS

At the top of the file, include a block of comments listing the name of the file, your name, the course and section number, the assignment number, and a brief description of the code’s purpose.

Also state in the comment block what famous object is represented in each image!
Here's a template:

/******************************************
* FileName.cpp
*
* Your Name Here
* Today's Date
* CS2308-00X – Project #Y
* Briefty describe what the code does.
* puzzle1: What is the image in solution1?
* puzzle2: What is the image in solution2?
* puzzle3: What is the image in solution3?
******************************************/

You must name your C++ source file as follows:

imagePuzzles_XXXXXXXXX.cpp

where XXXXXXXXX is your TxState ID number. (Note that your Texas State ID has the format A00123456, and it is NOT your NetID!)

Upload your C++ source file (named as above!) to TRACS.

TRACS will not allow submission after the deadline, so I strongly recommend that you don’t come down to the final seconds of the assignment window.

You do not need to submit any of your solution images.

No hard-copy turn-in is required for this assignment.

This assignment was adapted with my gratitude from a nifty assignment by Nick Parlante at Stanford.