Final Project: DIP – Discrete Image Processing

A matrix is a set of values arranged in rows and columns. Therefore, every element of a matrix has a row position and a column position and the matrix can be represented in C++ as a two-dimensional (2-d) array. An image file format, like a bitmap or pixmap, can be modeled by one such matrix. With the former, each element (or bit pixel) has a value of either one or zero whereas with the latter, each element (or pixel) can have several values; in an RGB format pixmap file, each pixel has three values, for R, G, and B, respectively. Further, RGB images can be represented in C++ as three [parallel] 2-d arrays where each array expresses the intensity of each color (red, green, or blue) in the combination for the pixels. The combination of these values defines the color of each pixel. A grayscale image is one such set of 2-d arrays where each of the three values (R, G, and B) of each pixel has the same value, which can be the average of its color counterpart. In an 24-bit deep color image, each of R, G, and B, can have 256 values (from 0 to 255).

There are different options for storing images in a file. PPM (Portable Pix Map) is a raster (e.g., raw) format. GIF and JPEG are compressed formats. In this program, you will be manipulating PPM files. The structure of a PPM file consists of two parts, a header and image data. In the header, the first line specifies the type of the image, `P6`; the second line shows the width and height of the image; the third line is the maximum intensity value. After the header follows the image data, arranged as `RGBRGBRGB...`, pixel by pixel in binary representation. Here is an example of a PPM image file:

```
P6
644 410
255
RGBRGBRGB...
```

In summary, you will write a program that reads an RGB .ppm image from the corresponding [given] image file; populates three 2-d arrays with the pixel values; performs one to a few image processing operations; and saves the arrays to another file. You can visualize .ppm images within the Virtual Box or other Linux installations by double-clicking on the file icon. However, you may need to convert the .ppm image to .jpg format in order to visualize the image using MS Windows or Mac OSX unless you install a .ppm viewer.

1. Write a header file, `finalproj.h`, which contains all the `#include`s, global and function declarations, etc for the project.
2. Write the following functions in their corresponding file (`*.cpp`):
   (a) `int menu()`, which interacts with the user to display the options and select the desired operation to perform.

(b) `int readImg(string, unsigned char[][HEIGHT], unsigned char[][HEIGHT], unsigned char[][HEIGHT]);` to read the image and populate the arrays.

(c) `int saveImg(string, unsigned char[][HEIGHT], unsigned char[][HEIGHT], unsigned char[][HEIGHT]);` to save the modified images.

(d) `int mirrorHoriz(unsigned char[][HEIGHT], unsigned char[][HEIGHT], unsigned char[][HEIGHT]);` to flip the image horizontally.

(e) `int mirrorVert(unsigned char[][HEIGHT], unsigned char[][HEIGHT], unsigned char[][HEIGHT]);` to flip the image vertically.

(f) `int negImg(unsigned char[][HEIGHT], unsigned char[][HEIGHT], unsigned char[][HEIGHT]);` to convert the image to its negative version.

3. Write a program, `finalproj.cpp` which contains the function `main()`

Example of interaction:

```
cs1st@cs1vm:~/finalproj$ make all
  g++ -c menu.cpp -o menu.o
  g++ -c readImg.cpp -o readImg.o
  g++ -c saveImg.cpp -o saveImg.o
  g++ -c mirrorHoriz.cpp -o mirrorHoriz.o
  g++ -c mirrorVert.cpp -o mirrorVert.o
  g++ -c negImg.cpp -o negImg.o
  g++ finalproj.cpp -o finalproj menu.o readImg.o saveImg.o
     mirrorHoriz.o mirrorVert.o negImg.o

cs1st@cs1vm:~/finalproj$ ./finalproj
Enter input file name: cs1a.ppm

1. Mirror horizontally
2. Mirror vertically
3. Convert to negative
4. exit

What would you like to do? 2

Image succesfully converted to negative and save in file cs1aneg.
   ppm

cs1st@cs1vm:~/finalproj$
```

Figure 1: original image

Figure 2: horizontal flip image

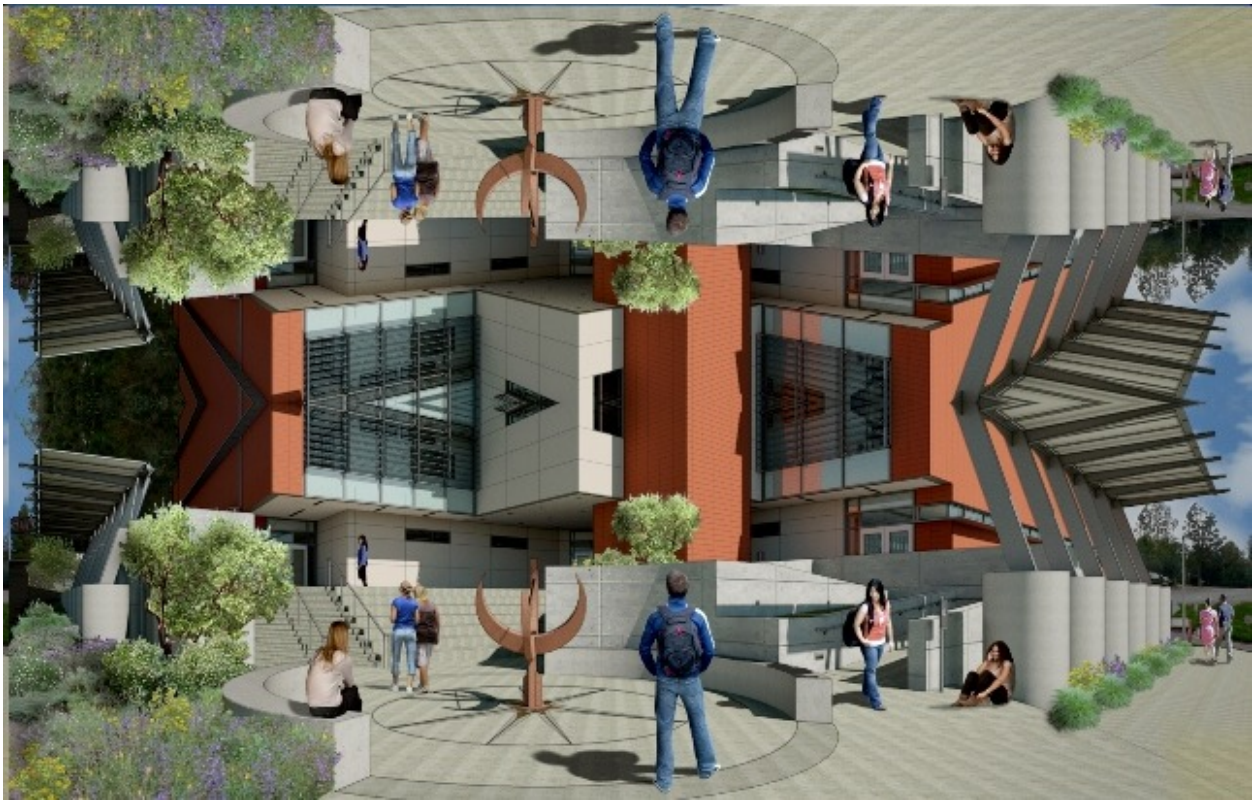

Figure 3: vertical flip image

Figure 4: negative image

Create a typescript output (using the command `script`) called `finalproj.scr`

Create a `tar` file called `finalproj.tar` containing the following files:

1. `finalproj.h`
2. `menu.cpp`
3. `readImg.cpp`
4. `saveImg.cpp`
5. `mirrorHoriz.cpp`
6. `mirrorVert.cpp`
7. `negImg.cpp`
8. `finalproj.cpp`
9. `finalproj.scr`

To create the `tar` file, use the following command:

```
tar cf finalproj.tar finalproj.h finalproj.cpp menu.cpp readImg.
   cpp saveImg.cpp mirrorHoriz.cpp mirrorVert.cpp negImg.cpp
   Makefile finalproj.scr
```

Submit the `tar` archive file `finalproj.tar` to canvas by the date on top of this page.