PS1b: PhotoMagic

Due: Monday, May 23, 11:59pm

In part (a), you wrote a program that produces pseudo-random bits by simulating a linear feedback shift register. Now, you will use it to implement a simple form of encryption for digital pictures.

For this portion of the assignment, you will:

- Read three arguments from the command line: source image filename, output image filename, and FibLFSR seed.
- Use SFML to load the source image from disk and display it in its own window.
- Use your debugged FibLFSR class to encode (or decode) the image.
- Display the encoded/decoded image in its own window.
- Save the new image to disk.

Average time to complete assignment: $\sim$ 6 hours.

1 Details

Please see starter code `pixels.cpp` for using SFML to load, manipulate, display, and save an image.

The code reads in a file and then uses individual pixel access to photographically negate an upper 200 px (pixels) square, like this:

Your task is to write a program PhotoMagic.cpp that can encrypt and decrypt pictures by implementing the following API:

```
// Transforms image using FibLFSR
void transform(sf::Image&, FibLFSR*);
// Display an encrypted copy of the picture, using the LFSR to do the
   encryption
```

Transform Function

The `transform()` function takes an image and a `FibLFSR` as arguments. It transforms the image using the linear feedback shift register as follows:

For each pixel (x, y) in row-major order - $(0, 0), (0, 1), (0, 2), \ldots$ - extract the red, green, and blue components of the color (each component is an integer between 0 and 255). Then XOR the red component with a newly-generated 8-bit integer. Do the same for the green (using another new 8-bit integer), and finally the blue. Create a new color using the result of the XOR operations, and set the pixel in the new picture to that color.

Main function

The `main()` function takes three command-line arguments: an input picture filename, an output picture filename, and a binary password (the initial LFSR seed). It should display the transformed picture on the screen.

Your main code should be in a file named `PhotoMagic.cpp` and should accept command line arguments as follows (e.g.):

```
% PhotoMagic input-file.png output-file.png 1011011000110110
```

which should take the input file and encrypt it using the method described above, with `FibLFSR` seed `1011011000110110`.

Your program should display the source file and encrypted file, and write out the encrypted file to `output-file.png`. Note: If you save as a `.jpg`, you won't be able to restore the file properly. (Why not?)

Then, if you re-run your program on the encrypted file, and give it the same LFSR seed, it should produce the original input file! Make sure you understand why.

Make sure to transform the **whole** image, not just the upper-left 200 × 200 pixels square from the demo. You can have 1 or 2 output windows.

Note: to work with two SFML windows, create two window objects (e.g. `window1` and `window2`). You can use this as your event loop:

```
while (window1.isOpen() && window2.isOpen()) {
    sf::Event event;
    while (window1.pollEvent(event)) {
        if (event.type == sf::Event::Closed)
            window1.close();
    }
    while (window2.pollEvent(event)) {
        if (event.type == sf::Event::Closed)
            window2.close();
    }
    window1.clear();
    window1.draw(/* fill in here */);
    window1.display();
    window2.clear();
    window2.draw(/* fill in here */);
    window2.display();
}
```

2 Extra Credit

If you're looking for a bigger challenge, consider converting from an alphanumeric password to the LFSR initial seed. If you do any of the extra credit work, make sure to describe exactly what you did in `Readme-ps1b.md`.

3 What to turn in

Your makefile should build two programs, `PhotoMagic` and `test`. Your `pname` program will perform the image transformation described above and your `test` program will run the unit tests you created in part (a) as well as any additional tests you created for part (b).

Submit a tarball to Blackboard containing:

- Your new code in `PhotoMagic.cpp`.
- Your code from part (a), including `FibLFSR.cpp`, `FibLFSR.hpp`, and `test.cpp`.
- The makefile for your project. The makefile should have targets `all`, `PhotoMagic`, `test` and `clean`. Make sure that all prerequisites are correct.
- Your `Readme-ps1b.md` that includes
  1. Your name
  2. Statement of functionality of your program (e.g. fully works, partial functionality, extra credit)
  3. Any other notes
- Any other source files that you created.
- Two screenshots; one showing a run where you encode the image and one where you decode the encrypted imaged.

Make sure that all of your files are in a directory named `ps1b` before archiving it and that there are no `.o` or other compiled files in it.

4 Grading rubric

Feature	Points	Comment
Core Implementation	8	Full & Correct Implementation
	2	Reads and writes the required files
	2	Encodes image
	2	Encoding is correct
	2	Displays both before and after image
Makefile	2	
	1	Has targets `PhotoMagic`, `test`, `all`, and `clean`
	1	`all` builds `PhotoMagic` and `test`
Screenshot	2	
	1	Two screenshots: encryption and decryption runs
	1	Both before and after images shown in each screenshot
Readme	2	Complete
Extra Credit	2	
	+1	Reads in an alphanumeric password and mentions in the readme that it does this
	+1	Readme describes how they convert the password into a bit sequence
Penalties		
	-1	Submission includes `.o` files
	-1	Submission not in a directory named `ps1b`
	-10%	Each day late
Total	14	