```
    1: /**
    2: * photoMagic.cpp - Essentially to encode and decode an image using the Fi
bLFSR
    3: * that was programmed back in ps1a as an assignment
    4: *
    5: * Date 2/1/22 - 2/7/22
    6: *
    7: * Created by: Anson Cheang
    8: *
    9: */
   10:
   11: #include <SFML/System.hpp>
   12: #include <SFML/Window.hpp>
   13: #include <SFML/Graphics.hpp>
   14: #include "FibLFSR.h"
   15:
   16: // transforms image using FibLFSR
   17: void transform( sf::Image&, FibLFSR*);
   18:
   19: // display an encrypted copy of the picture, using the LFSR
   20: // to do the encryption
   21: int main(int argc, char* argv[])
   22: {
   23:     sf::Image image1;
   24:     FibLFSR encryptionCode(argv[3]);
   25:         if (!image1.loadFromFile(argv[1]))
   26:     {
   27:                 return -1;
   28:     }
   29:
   30:         sf::Vector2u size = image1.getSize();
   31:         sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "Input");
   32:
   33:         sf::Texture texture;
   34:         texture.loadFromImage(image1);
   35:
   36:         sf::Sprite sprite;
   37:         sprite.setTexture(texture);
   38:
   39:     transform(image1, &encryptionCode);
   40:
   41:     sf::Vector2u size2 = image1.getSize();
   42:         sf::RenderWindow window2(sf::VideoMode(size2.x, size2.y), "Output
");
   43:
   44:         sf::Texture texture2;
   45:         texture2.loadFromImage(image1);
   46:
   47:         sf::Sprite sprite2;
   48:         sprite2.setTexture(texture2);
   49:
   50:     while (window1.isOpen() && window2.isOpen())
   51:     {
   52:         sf::Event event;
   53:         while (window1.pollEvent(event)) {
   54:             if (event.type == sf::Event::Closed)
   55:             {
   56:                 window1.close();
   57:             }
   58:         }
   59:         while (window2.pollEvent(event))
   60:         {
   61:             if (event.type == sf::Event::Closed)
   62:             {
   63:                 window2.close();
```

```
64:             }
65:         }
66:         window1.clear();
67:         window1.draw(sprite);
68:         window1.display();
69:         window2.clear();
70:         window2.draw(sprite2);
71:         window2.display();
72:     }
73:
74:     if (!image1.saveToFile(argv[2]))
75:         {
76:         return -1;
77:     }
78:
79:     return 0;
80: }
81:
82: void transform( sf::Image& image, FibLFSR* encryptionCode)
83: {
84:     sf::Color p;
85:         sf::Vector2u size = image.getSize();
86:
87:         // create photographic negative image of upper-left 200 px square
88:         for (int x = 0; x < static_cast<int>(size.x); x++) {
89:                 for (int y = 0; y < static_cast<int>(size.y); y++) {
90:                         p = image.getPixel(x, y);
91:                         p.r = p.r ^ encryptionCode->generate(8);
92:                         p.g = p.g ^ encryptionCode->generate(8);
93:                         p.b = p.b ^ encryptionCode->generate(8);
94:                         image.setPixel(x, y, p);
95:                 }
96:         }
97: }
```