```
    1: /*
    2: Computing IV - Assignment - PS1a + b
    3: Instructor: Prof. Yelena Rykalova
    4: Due Date: 02/07/22
    5: Author: Matthew Lorette Anaya
    6: Description: This program is an implementation of a Fibonacci Linear Feed
back                Shift Register
    7:              This is the implementation of the PhotoMagic.class which tak
es thre             e arguments an input image an output image and a seed. The p
rogram              uses the the seed to encode the input image and display it a
s the o             utput image.
    8: */
    9: #include <iostream>
   10: #include <string>
   11: #include <sstream>
   12: #include <SFML/System.hpp>
   13: #include <SFML/Window.hpp>
   14: #include <SFML/Graphics.hpp>
   15: #include "FibLFSR.h"
   16:
   17: void transform( sf::Image& img, FibLFSR* bit_generator) {
   18:   // randomize the bits in the image
   19:   sf::Vector2u imgsize = img.getSize();
   20:   // initialize an SFML pixel
   21:   sf::Color p;
   22:
   23:   for(int x = 0; x < (signed)imgsize.x; x++) {
   24:     for(int y = 0; y < (signed)imgsize.y; y++) {
   25:       // get the current pixel from the input image
   26:       p = img.getPixel(x, y);
   27:
   28:       // generate encoded pixels
   29:       p.r = p.r ^ bit_generator -> generate(8);
   30:       p.g = p.g ^ bit_generator -> generate(8);
   31:       p.b = p.b ^ bit_generator -> generate(8);
   32:
   33:     // edit the image in-place with new encoded pixels
   34:       img.setPixel(x, y, p);
   35:     }
   36:   }
   37: }
   38: int main(int argc, char* argv[]) {
   39:   if(argc != 4) {
   40:     std::cout << "Incorrect Input Format" << std::endl
   41:               << "Input should be as follows: ./PhotoMagic <inputfilename
> <outputfilename> <seed>\n";
   42:     return -1;
   43:   }
   44:
   45:   // store input in variables
   46:   std::string input_fname(argv[1]);
   47:   std::string output_fname(argv[2]);
   48:   std::string seed = argv[3];
   49:
   50:   // create an LSFR object
   51:   FibLFSR bit_generator(seed);
   52:
   53:   // load images
   54:   sf::Image input_image;
   55:   if (!input_image.loadFromFile(input_fname)) {
   56:       return -1;
   57:   }
   58:
   59:   sf::Image output_image;
   60:   if (!output_image.loadFromFile(input_fname)) {
```

```
   61:       return -1;
   62:   }
   63:
   64:   // display 2 windows
   65:   sf::Vector2u imgsize = input_image.getSize();
   66:   sf::RenderWindow input_window(sf::VideoMode(imgsize.x, imgsize.y), "Inp
ut Image");
   67:   sf::RenderWindow output_window(sf::VideoMode(imgsize.x, imgsize.y), "Ou
tput Image");
   68:
   69:   // load the images into textures
   70:   sf::Texture in_texture, out_texture;
   71:   in_texture.loadFromImage(input_image);
   72:
   73:   transform(input_image, &bit_generator);
   74:
   75:   out_texture.loadFromImage(input_image);
   76:
   77:   // load textures -> sprites
   78:   sf::Sprite in_sprite, out_sprite;
   79:   in_sprite.setTexture(in_texture);
   80:   out_sprite.setTexture(out_texture);
   81:
   82:   // main loop
   83:   while (input_window.isOpen() && output_window.isOpen()) {
   84:       sf::Event event;
   85:
   86:       while (input_window.pollEvent(event)) {
   87:           if (event.type == sf::Event::Closed) {
   88:               input_window.close();
   89:             }
   90:         }
   91:
   92:       while (output_window.pollEvent(event)) {
   93:           if (event.type == sf::Event::Closed) {
   94:               output_window.close();
   95:             }
   96:         }
   97:
   98:       input_window.clear();
   99:       input_window.draw(in_sprite);    // Input image
  100:       input_window.display();
  101:
  102:       output_window.clear();
  103:       output_window.draw(out_sprite);    // Output image
  104:       output_window.display();
  105:     }
  106:
  107:   // save the image
  108:   if (!input_image.saveToFile(output_fname)) {
  109:       return -1;
  110:     }
  111:
  112:   return 0;
  113: }
```