```
    1: #include <SFML/System.hpp>
    2: #include <SFML/Window.hpp>
    3: #include <SFML/Graphics.hpp>
    4: #include <string>
    5: #include <cstdlib>
    6: #include "LFSR.hpp"
    7:
    8: void transform_method(sf::Image& image, sf::Color p, sf::Vector2u size, LFSR
 l);
    9:
   10: int main(int argc, char* argv[])
   11: {
   12:         //Read in Command Line Arguments
   13:         std::string user_image1 = argv[1];
   14:         std::string user_image2 = argv[2];
   15:         int tap = atoi(argv[4]);
   16:         LFSR l(argv[3], tap);
   17:
   18:         //Load Images
   19:         sf::Image image;
   20:         sf::Image image1;
   21:         if (!image.loadFromFile(user_image1))
   22:           {
   23:             return -1;
   24:           }
   25:         if (!image1.loadFromFile(user_image1))
   26:           {
   27:             return -1;
   28:           }
   29:
   30:         //Create pixel variable
   31:         sf::Color p;
   32:
   33:         //Set Size
   34:         sf::Vector2u size = image1.getSize();
   35:
   36:         //Transform the Images
   37:         transform_method(image1, p, size, l);
   38:
   39:         //Save Image
   40:         if (!image1.saveToFile(user_image2))
   41:           {
   42:             return -1;
   43:           }
   44:
   45:         //After transform
   46:         sf::Image image2;
   47:         if (!image2.loadFromFile(user_image2))
   48:           {
   49:             return -1;
   50:           }
   51:
   52:         //Generate Windows
   53:         sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "Input");
   54:         sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "Output");
   55:
   56:         //Load Textures
   57:         sf::Texture texture1;
   58:         texture1.loadFromImage(image);
   59:         sf::Texture texture2;
   60:         texture2.loadFromImage(image2);
```

```
   61:
   62:         //Set Sprites
   63:         sf::Sprite sprite1;
   64:         sprite1.setTexture(texture1);
   65:
   66:         sf::Sprite sprite2;
   67:         sprite2.setTexture(texture2);
   68:
   69:
   70:
   71:         //Generate Windows
   72:         while (window1.isOpen() && window2.isOpen())
   73:         {
   74:                 sf::Event event;
   75:                 while (window1.pollEvent(event))
   76:                 {
   77:                     if (event.type == sf::Event::Closed)
   78:                       {
   79:                         window1.close();
   80:                       }
   81:                 }
   82:
   83:                 while (window2.pollEvent(event))
   84:                 {
   85:                     if (event.type == sf::Event::Closed)
   86:                       {
   87:                         window2.close();
   88:                       }
   89:                 }
   90:
   91:                 window1.clear(sf::Color::White);
   92:                 window1.draw(sprite1);
   93:                 window1.display();
   94:
   95:                 window2.clear(sf::Color::White);
   96:                 window2.draw(sprite2);
   97:                 window2.display();
   98:         }
   99:
  100:         return 0;
  101: }
  102:
  103: void transform_method(sf::Image& image, sf::Color p, sf::Vector2u size, LFSR
 l)
  104: {
  105:   for (unsigned int x = 0; x<size.x; x++)
  106:   {
  107:     for (unsigned int y = 0; y< size.y; y++)
  108:       {
  109:         int new_bit1, new_bit2, new_bit3;
  110:
  111:         new_bit1 = l.generate(8);
  112:         new_bit2 = l.generate(8);
  113:         new_bit3 = l.generate(8);
  114:
  115:         p = image.getPixel(x, y);
  116:         p.r = p.r ^ new_bit1;
  117:         p.g = p.g ^ new_bit2;
  118:         p.b = p.b ^ new_bit3;
  119:         image.setPixel(x, y, p);
  120:       }
```

```
121:    }
122: }
```

```
 1: #ifndef LFSR_HPP
 2: #define LFSR_HPP
 3:
 4: #include <iostream>
 5: #include <string>
 6:
 7: class LFSR
 8: {
 9:
10: public:
11:   LFSR(std::string user_seed, int user_tap);
12:   ~LFSR();
13:   int step();
14:   int generate(int k);
15:
16:   friend std::ostream& operator<< (std::ostream &out, LFSR &lfsr);
17:
18: private:
19:   std::string seed;
20:   int tap;
21:
22: };
23:
24:
25: #endif
```

```
 1: #include "LFSR.hpp"
 2:
 3: LFSR::LFSR(std::string user_seed, int user_tap)
 4: {
 5:   tap = user_tap;
 6:   seed = user_seed;
 7: }
 8:
 9: LFSR::~LFSR()
10: {
11:
12: }
13:
14: int LFSR::step()
15: {
16:
17:   int bit;
18:   char c_bit;
19:
20:   if (seed.at(0) == seed.at((seed.size()-1) - tap))
21:     {
22:       bit = 0;
23:       c_bit = '0';
24:     }
25:   else
26:     {
27:       bit = 1;
28:       c_bit = '1';
29:     }
30:
31:   for(unsigned int i = 0; i < (seed.size()-1); i++)
32:     {
33:       seed.at(i) = seed.at(i+1);
34:     }
35:   seed.at(seed.size()-1) = c_bit;
36:
37:   return bit;
38: }
39:
40: int LFSR::generate(int k)
41: {
42:   int gen = 0;
43:
44:   for (int i =0; i < k; i++)
45:     {
46:       gen = gen*2 + LFSR::step();
47:     }
48:
49:   return gen;
50: }
51:
52: std::ostream& operator<<(std::ostream &out,LFSR &lfsr)
53: {
54:   out << lfsr.seed;
55:
56:   return out;
57: }
```

```
 1: C=g++ -g -Wall --std=c++98 -Werror
 2: E=.cpp
 3: O=PhotoMagic.o LFSR.o
 4: P=PhotoMagic
 5: BOOST= -lboost_unit_test_framework
 6: SFML= -lsfml-graphics -lsfml-window -lsfml-system
 7: all: $(P)
 8: $(P):$(O)
 9:         $(C) -o $(P) $(O) $(BOOST) $(SFML)
10:
11: $(E).o:
12:         $(C) -c $<  -o $@
13:
14: clean:
15:         rm $(O) $(P)
```