```
 1: #include <SFML/System.hpp>
 2: #include <SFML/Window.hpp>
 3: #include <SFML/Graphics.hpp>
 4: #include <iostream>
 5: #include <string>
 6: #include <algorithm>
 7: #include "LFSR.hpp"
 8:
 9: sf::Image transform(sf::Image picture, LFSR lfsr)
10: {
11:   sf::Vector2u size = picture.getSize();
12:   sf::Color p2;
13:   for (size_t x = 0; x < size.x; x++) {
14:     for (size_t y = 0; y < size.y; y++) {
15:       p2 = picture.getPixel(x, y);
16:       p2.r = p2.r ^ lfsr.generate(8);
17:       p2.g = p2.g ^ lfsr.generate(8);
18:       p2.b = p2.b ^ lfsr.generate(8);
19:       picture.setPixel(x, y, p2);
20:     }
21:   }
22:   return picture;
23: }
24:
25: int main(int argc, char *argv[])
26: {
27:   std::string fi, out;
28:   if (argc < 3) {
29:     std::cout << "Usage: " << argv[0] << " inputfile LFSR bit" << std::endl;
30:     exit(0);
31:   } else {
32:     fi = argv[1];
33:     out = argv[2];
34:   }
35:   sf::Image image;
36:   if (!image.loadFromFile(fi))
37:     return -1;
38:   sf::Image imageOut;
39:   if (!imageOut.loadFromFile(fi))
40:     return -1;
41:   LFSR lfsr(argv[3], atoi(argv[4]));
42:   imageOut = transform(imageOut, lfsr);
43:   sf::Vector2u size = image.getSize();
44:   sf::RenderWindow window(sf::VideoMode(size.x * 2, size.y), "Meow");
45:   if (!imageOut.saveToFile(fi))
46:     return -1;
47:   sf::Texture texture;
48:   texture.loadFromImage(image);
49:   sf::Sprite sprite;
50:   sprite.setTexture(texture);
51:   sf::Texture textureOut;
52:   textureOut.loadFromImage(imageOut);
53:   sf::Sprite spriteOut;
54:   spriteOut.setTexture(textureOut);
55:   spriteOut.setPosition(size.x, 0);
56:   while (window.isOpen()) {
57:     sf::Event event;
58:     while (window.pollEvent(event)) {
59:       if (event.type == sf::Event::Closed)
60:         window.close();
61:     }
```

```
   62:     window.clear(sf::Color::White);
   63:     window.draw(sprite);
   64:     window.draw(spriteOut);
   65:     window.display();
   66:   }
   67:
   68:
   69:
   70:   if (!imageOut.saveToFile(out))
   71:     return -1;
   72:   return 0;
   73: }
```

```
 1: #include <iostream>
 2: #include <cmath>
 3: #include "LFSR.hpp"
 4:
 5: LFSR::LFSR(std::string seed, int t)
 6: {
 7:     _data = seed;
 8:     length = _data.size();
 9:     tap = t;
10: }
11:
12: int LFSR::step()
13: {
14:     int bit;
15:     int length = _data.size();
16:     std::string s_bit;
17:     bit = _data.front() ^ _data[length - tap - 1];
18:     s_bit = std::to_string(bit);
19:     _data.erase(0,1);
20:     _data = _data + s_bit;
21:     return bit;
22: }
23:
24: int LFSR::generate(int k)
25: {
26:     int count = 0;
27:     for (int i = k - 1 ;i >= 0; i--)
28:     {
29:         if(step() == 1)
30:             count+= pow(2, i);
31:     }
32:     return count;
33: }
34:
35: std::ostream& operator << (std::ostream& out, LFSR& lfsr)
36: {
37:     out << lfsr._data;
38:     return out;
39: }
```