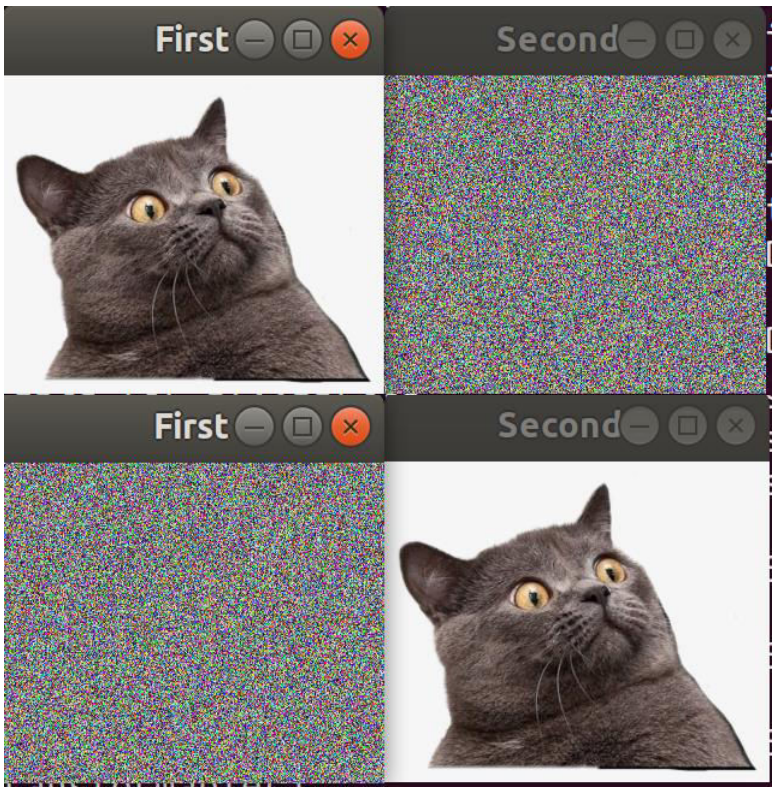
PS1 Linear Feedback Shift Register

In this assignment, I had to encrypt an image using the LFSR generate command and save that image, then decrypt that image to get the original image using the same LFSR generate command. OO designs that were central to this assignment were classes. An additional implementation that I added to this assignment was that the esc key would close the window.

There was only serious problem I encountered was that I copied and pasted p.r ^ lfsr.generate(5) for the p.g and p.b and it took me forever to find out what was wrong with it, and the make file.



```
 1: CC = g++
 2: CFLAGS = -std=c++11 -c -g -Wall -Werror -pedantic
 3: LIBS = -lsfml-graphics -lsfml-window -lsfml-system
 4:
 5: all: PhotoMagic
 6:
 7: PhotoMagic: PhotoMagic.o LFSR.o
 8:         $(CC) PhotoMagic.o LFSR.o -o PhotoMagic $(LIBS)
 9:
10: PhotoMagic.o: PhotoMagic.cpp LFSR.cpp
11:         $(CC) -c PhotoMagic.cpp LFSR.cpp
12:
13: LFSR.o: LFSR.cpp LFSR.hpp
14:         $(CC) $(CFLAGS) LFSR.cpp -o LFSR.o
15:
16: clean:
17:         rm *.o PhotoMagic
```

```
 1: //#include <iostream>
 2: //#include <string>
 3: #include "LFSR.hpp"
 4: using namespace std;
 5:
 6: LFSR::LFSR(string seed, int tap) {
 7:         for (unsigned int i = 0; i < seed.length(); i++)
 8:                 this->seed.push_back(seed[i]);
 9:         //save_seed = seed;
10:         //since the tap is counted from right to left, must take total lengt
h
11:         //and subtract it from the input tap
12:         this->tap = seed.length() - tap - 1;
13: }
14:
15: int LFSR::step() {
16:         int first = seed.at(0),
17:             _tap = seed.at(tap),
18:                 n_bit = first ^ _tap;
19:         seed.erase(seed.begin());
20:         seed.push_back(n_bit);
21:         return n_bit;
22: }
23:
24: int LFSR::generate(int k) {
25:         int val, output = 0;
26:         for (int i = 0; i < k; i++) {
27:                 val = step();
28:                 output = (output * 2) + val;
29:         }
30:         return output;
31: }
32:
33: ostream& operator<< (ostream &out, const LFSR &obj) {
34:         for (unsigned int i = 0; i < obj.seed.size(); i++) {
35:                 out << obj.seed[i];
36:         }
37:
38:         return out;
39: }
40:
41: //LFSR::~LFSR() {}
```

```
 1: #include <iostream>
 2: #include <string.h>
 3: #include <vector>
 4: using namespace std;
 5:
 6: class LFSR
 7: {
 8: public:
 9:         LFSR(string seed, int tap);
10:
11:         int step();
12:
13:         int generate(int k);
14:
15:         friend ostream& operator<< (ostream &out, const LFSR &obj);
16:
17:         //~LFSR();
18:
19: private:
20:         vector<int> seed;
21:         int tap;
22: };
```

```
    1: // pixels.cpp:
    2: // using SFML to load a file, manipulate its pixels, write it to disk
    3: // Fred Martin, fredm@cs.uml.edu, Sun Mar  2 15:57:08 2014
    4:
    5: // g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window
    6:
    7: #include <SFML/System.hpp>
    8: #include <SFML/Window.hpp>
    9: #include <SFML/Graphics.hpp>
   10: #include "LFSR.hpp"
   11:
   12:
   13: int main(int argc, char* argv[])
   14: {
   15:         string input_file = argv[1];
   16:         string output_file = argv[2];
   17:         string seed = argv[3];
   18:         int tap = atoi(argv[4]);
   19:
   20:         sf::Image first;
   21:         if (!first.loadFromFile(input_file))
   22:                 return -1;
   23:
   24:         sf::Image second;
   25:         if (!second.loadFromFile(input_file))
   26:                 return -1;
   27:
   28:         // p is a pixel
   29:         sf::Color p;
   30:         sf::Vector2u win1_size = first.getSize();
   31:
   32:         LFSR lfsr(seed, tap);
   33:
   34:         // create encrypted image of the original image
   35:         for (unsigned int x = 0; x < win1_size.x; x++) {
   36:                 for (unsigned int y = 0; y < win1_size.y; y++) {
   37:                         p = second.getPixel(x, y);
   38:                         p.r = p.r ^ lfsr.generate(5);
   39:                         p.g = p.g ^ lfsr.generate(5);
   40:                         p.b = p.b ^ lfsr.generate(5);
   41:                         second.setPixel(x, y, p);
   42:                 }
   43:         }
   44:
   45:         sf::RenderWindow window1(sf::VideoMode(win1_size.x, win1_size.y), "F
irst");
   46:         sf::RenderWindow window2(sf::VideoMode(win1_size.x, win1_size.y), "S
econd");
   47:
   48:         sf::Texture original;
   49:         original.loadFromImage(first);
   50:         sf::Texture encrypted;
   51:         encrypted.loadFromImage(second);
   52:
   53:         sf::Sprite sprite1;
   54:         sprite1.setTexture(original);
   55:         sf::Sprite sprite2;
   56:         sprite2.setTexture(encrypted);
   57:
   58:         while (window1.isOpen() && window2.isOpen()) {
   59:         sf::Event event;
```

```
   60:                   while (window1.pollEvent(event)) {
   61:                           if (event.type == sf::Event::Closed)
   62:                                   window1.close();
   63:                           else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Es
cape))
   64:                                   window1.close();
   65:                   }
   66:                   while (window2.pollEvent(event)) {
   67:                           if (event.type == sf::Event::Closed)
   68:                                   window2.close();
   69:                           else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Es
cape))
   70:                                   window2.close();
   71:                           }
   72:                   window1.clear();
   73:                   window1.draw(sprite1);
   74:                   window1.display();
   75:                   window2.clear();
   76:                   window2.draw(sprite2);
   77:                   window2.display();
   78:           }
   79:
   80:           // fredm: saving a PNG segfaults for me, though it does properly
   81:           //   write the file
   82:           if (!second.saveToFile(output_file))
   83:                   return -1;
   84:
   85:           return 0;
   86: }
```