```
    1: all: PhotoMagic
    2:
    3: PhotoMagic: PhotoMagic.o LFSR.o
    4:         g++ PhotoMagic.o LFSR.o -o PhotoMagic -lsfml-graphics -lsfml-window
-lsfml-system
    5:
    6: PhotoMagic.o: PhotoMagic.cpp LFSR.hpp
    7:         g++ -c PhotoMagic.cpp
    8:
    9: LFSR.o: LFSR.cpp LFSR.hpp
   10:         g++ -c LFSR.cpp
   11: clean: rm *.o PhotoMagic *~
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2b
 3: /*#include <iostream>
 4: #include <string>
 5: #include "LFSR.hpp"
 6: using namespace std ;
 7: int main( int argc, char *argv[] ) {
 8:
 9:   int amount ;
10:   LFSR *test  = new LFSR( "1101", 2 ) ;
11:
12:   //amount = test->step() ;
13:   //cout << *test << endl ;
14:   //cout << amount << endl ;
15:
16:   amount = test->generate(3) ;
17:   cout << amount << endl ;
18:   cout << *test << endl  ;
19:   delete test ;
20: } */
21:
22: // pixels.cpp:
23: // using SFML to load a file, manipulate its pixels, write it to disk
24: // Fred Martin, fredm@cs.uml.edu, Sun Mar  2 15:57:08 2014
25:
26: // g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window
27:
28: #include <SFML/System.hpp>
29: #include <SFML/Window.hpp>
30: #include <SFML/Graphics.hpp>
31: #include "LFSR.hpp"
32: int main(int argc, char *argv[] ) {
33:   int n = atoi(argv[4]) ;
34:   LFSR *pow = new LFSR( argv[3], n  ) ;
35:   sf::Image image;
36:   sf::Image image2;
37:   if (!image.loadFromFile(argv[1]))  return -1;
38:   if (!image2.loadFromFile(argv[1])) return -1 ;
39:   // p is a pixel
40:   sf::Color p;
41:   sf::Vector2u size = image.getSize();
42:
43:   for (int x=0; (unsigned) x< size.x; x++) {
44:     for (int y= 0;(unsigned) y< size.y; y++) {
45:       p = image.getPixel(x, y);
46:       p.r =  p.r ^ pow->generate(8) ;
47:       p.g =  p.g ^ pow->generate(8) ;
48:       p.b =  p.b ^ pow->generate(8);
49:       image.setPixel(x, y, p);
50:     }
51:   }
52:
53:   sf::RenderWindow window(sf::VideoMode(size.x, size.y), "Angel C. LFSR");
54:   sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "Angel C. LFSR");
55:   sf::Texture texture;
56:   sf::Texture t2;
57:
58:   texture.loadFromImage(image);
59:   t2.loadFromImage(image2) ;
60:
61:   sf::Sprite sprite;
```

```
   sf::Sprite s2 ;

   sprite.setTexture(texture);
   s2.setTexture(t2 ) ;

   while (window.isOpen() && window2.isOpen()){
       sf::Event event;
       while (window.pollEvent(event)) {
           if (event.type == sf::Event::Closed)
             window.close();
         }
         while (window2.pollEvent(event)) {
           if (event.type == sf::Event::Closed)
             window.close();
         }
       window.clear();
       window.draw(sprite);
       window.display();
       window2.clear();
       window2.draw(s2);
       window2.display();
     }
   if (!image.saveToFile(argv[2]))
     return -1;
delete pow ;
   return 0;
}
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2b
 3: #ifndef _LFSR
 4: #define _LFSR
 5:
 6: #include <iostream>
 7: #include <string>
 8:
 9: using namespace std ;
10:
11: class LFSR{
12: protected:
13:   string unit ; int t, bit ;
14:
15: public :
16:   LFSR( string seed, int tap ) : unit(seed), t(tap) {} ;
17:   int step() ;
18:   int generate( int k ) ;
19:   int stTodig( string &a ) ;
20:
21:   friend std::ostream& operator<<( std::ostream &out , const LFSR& rhs) {
22:     out << rhs.unit ;
23:     return out ;
24:   }
25:   string prtln( const LFSR &t, int b ) ;
26: } ;
27:
28: #endif
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2b
 3: #include <iostream>
 4: #include <string>
 5: #include "LFSR.hpp"
 6:
 7: using namespace std ;
 8:
 9: int LFSR::stTodig( string &a ) {
10:   char b; int d1, d2 ;
11:
12:   b = a[0] ;
13:   if( b == '1' ) d1 = 1 ;
14:   else d1 = 0 ;
15:
16:   b = a[a.length() - t - 1] ;
17:   if( b == '1' ) d2 = 1;
18:   else d2 = 0 ;
19:
20:   return d1 ^ d2 ;
21: }
22: //unit[unit.length() - t - 1]
23: int LFSR::step(){
24:   char c ;
25:   //stTodig(untit)
26:   bit = stTodig( unit ) ;
27:   c = (char)bit ;
28:   unit.erase(unit.begin()) ;
29:   unit.append(1,c) ;
30:   return bit ;
31: }
32:
33: int LFSR::generate( int k ) {
34: // calls step k times. on the kth time take
35: //the string and take the k amount of bits
36: // and return it's value
37:
38:   int i, num, total ;
39:
40:   total = 0 ;
41:
42:   for( i = 0 ; i < k ; i++ )
43:     step() ;
44:   for( i = 0 ; i < k ; i++ ) {
45:     if( i == 0 ) num = 1 ;
46:     else num = num * num ;
47:     if( unit[i]== '1' ) total = total + num ;
48:   }
49:   return total ;
50: }
51: /*
52: std::ostream& operator<< ( std::ostream &out, LFSR& rhs ) {
53:
54:   out << rhs.unit ;
55:   return out ;
56: }
57: */
58: string LFSR::prtln(const LFSR &temp, int bit ) {
59:   cout << temp << " " << bit ;
60: }
61:
```

```
62:
63:
```