```
    1: all: ps2a
    2:
    3: ps2a: test.o LFSR.o
    4:         g++ test.o LFSR.o -o ps2a -lboost_unit_test_framework
    5:
    6: ps2a.o: test.cpp LFSR.hpp
    7:         g++ -c test.cpp -Wall -Werror -ansi -pedantic -lboost_unit_test_fram
ework -g
    8:
    9: LFSR.o: LFSR.cpp LFSR.hpp
   10:         g++ -c LFSR.cpp -Wall -Werror -ansi -pedantic -lboost_unit_test_fram
ework -g
   11:
   12: clean:
   13:         rm *.o ps2a *~ *.gch
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2a
 3: #define BOOST_TEST_DYN_LINK
 4: #define BOOST_TEST_MODULE Main
 5: #include <boost/test/unit_test.hpp>
 6: #include <iostream>
 7: #include <string>
 8: #include "LFSR.hpp"
 9:
10: BOOST_AUTO_TEST_CASE(fiveBitsTapAtTwo) {
11:   LFSR l("00111", 2);
12:
13:   BOOST_REQUIRE(l.step() == 1);
14:   BOOST_REQUIRE(l.step() == 1);
15:   BOOST_REQUIRE(l.step() == 0);
16:   BOOST_REQUIRE(l.step() == 0);
17:   BOOST_REQUIRE(l.step() == 0);  // boost finds an error here
18:   BOOST_REQUIRE(l.step() == 1);
19:   BOOST_REQUIRE(l.step() == 1);
20:   BOOST_REQUIRE(l.step() == 0);
21:   LFSR l2("00111", 2);
22:   BOOST_REQUIRE(l2.generate(8) == 198);
23: }
24:
25: BOOST_AUTO_TEST_CASE(sevenBitsTapAtfive) {
26:   LFSR s("1101010", 5);
27:   BOOST_REQUIRE(s.generate(4) != 102);
28: }
29:
30: BOOST_AUTO_TEST_CASE(sevenBitsTapAtfivea) {
31:   LFSR s("1101010", 5);
32:   BOOST_REQUIRE(s.step() == 0);
33:   BOOST_REQUIRE(s.step() == 1);
34:   BOOST_REQUIRE(s.step() == 1);
35:   BOOST_REQUIRE(s.step() == 1);
36:   BOOST_REQUIRE(s.step() == 1);
37:   std::stringstream sng;
38:   sng << s;
39:   BOOST_CHECK_EQUAL(sng.str(), "1001111");
40: }
41:
42: BOOST_AUTO_TEST_CASE(elevenbitsatseven) {
43:   LFSR l("11101000111", 7);
44:   BOOST_REQUIRE(l.step() == 1);
45:   BOOST_REQUIRE(l.step() == 0);
46:   BOOST_REQUIRE(l.step() == 1);
47:   BOOST_REQUIRE(l.step() == 0);
48:   BOOST_REQUIRE(l.step() == 1);
49:   LFSR l2("11101000111", 7);
50: }
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2a
 3:
 4: #ifndef _LFSR_
 5: #define _LFSR_
 6: #include <iostream>
 7: #include <string>
 8:
 9: class LFSR{
10:  protected:
11:   std::string unit; int t, bit;
12:  public :
13:   LFSR(std::string seed, int tap) : unit(seed), t(tap) {}
14:   int step();
15:   int generate(int k);
16:   int stTodig(std::string &a);
17:   friend std::ostream& operator<<(std::ostream &out , const LFSR& rhs) {
18:     out << rhs.unit;
19:     return out;
20:   }
21:   std::string prtln(const LFSR &t, int b);
22: };
23:
24: #endif
```

```
 1: // Copyright 2015 <Angel Z'heondre Calcano>
 2: // PS2a
 3: #include <iostream>
 4: #include <string>
 5: #include <vector>
 6: #include "LFSR.hpp"
 7:
 8: using namespace std;
 9:
10: int LFSR::stTodig(std::string &a) {
11:   char b; int d1, d2;
12:   b = a[0];
13:   if (b == '1') d1 = 1;
14:   else
15:     d1 = 0;
16:   b = a[a.length() - t - 1];
17:   if (b == '1') d2 = 1;
18:   else
19:     d2 = 0;
20:   return (d1 ^ d2);
21: }
22: int LFSR::step() {
23:   int bit; char c; char digits[3] = { '0', '1', '\0' };
24:   bit = stTodig(unit);
25:   c = digits[bit];
26:   unit.erase(unit.begin());
27:   unit.push_back(c);
28:   return bit;
29: }
30: int LFSR::generate(int k) {
31:   int i, num, total,count;
32:   vector< int > reg;
33:   reg.resize(k);
34:   total = count =0;
35:   num = 1;
36:   for(i = 0; i < k; i++) {
37:     reg[i] = step();
38:   }
39:   for ( i = k - 1; -1 < i; i--) {
40:     if( i != k - 1 )
41:       num = num * 2;
42:     if( reg[i] == 1){
43:       total += num;
44:     };
45:   }
46:   return total;
47: }
48: std::string LFSR::prtln(const LFSR &temp, int bit) {
49:   std::cout << temp << " " << bit;
50:   return "";
51: }
52:
53:
54:
```