Ps1a: FibLFSR

Assignment Description:

This assignment required us to make a pseudo-random number generator. The random number generator is based on a Linear feedback shift register or LFSR. This will generate random numbers based on a given key. These numbers are output to the terminal. These pseudo-random numbers are used in the next part of the assignment.

Key Concepts and Algorithms:

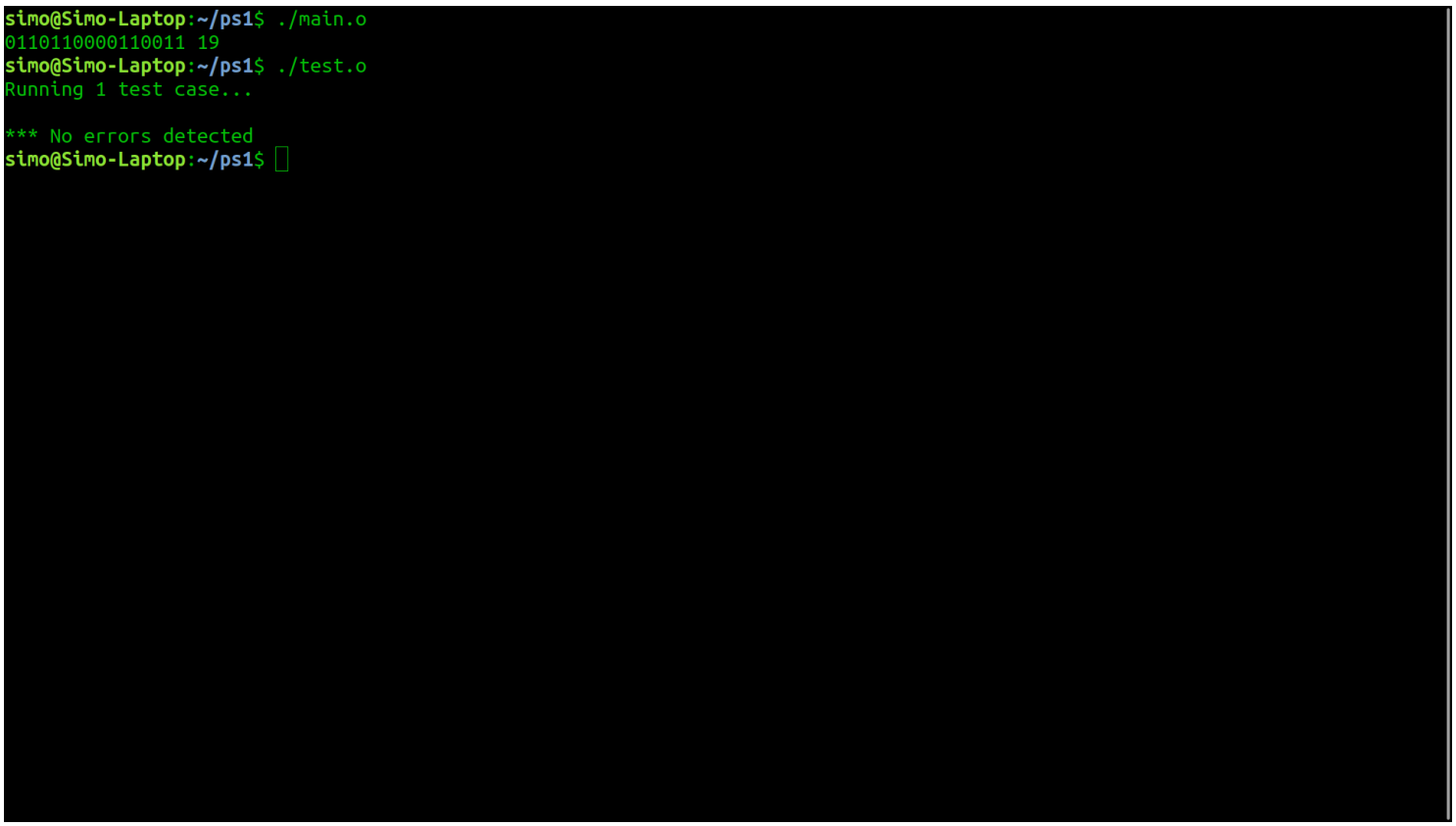
The main algorithm for this is a Linear feedback shift register which takes the last bit and XORs it the 3rd to last bit, then the 4th to last, and finally the 6th to last bit. This generated a pseudo-random number based on the 16 bit input key.

To accomplish the task of generating a random number we used a class called FibLFSR. This class holds all of the data and functions for generating the numbers. This is accomplished via the generate() and step() functions.

What I learned in this assignment:

In this assignment I learned about how to generate pseudo-random numbers using C++. I learned about classes in Computing III and how the XOR operation works in Computing I and Logic Design. This assignment wasn't too difficult and mostly served to lay the ground work for the next part of this assignment.

Ps1a: Screen shot



Ps1a Source Code: Makefile

```
CC = g++
CFLAGS =    -Wall -Werror
DEPS =      FibLFSR.hpp

main: main.o FibLFSR.o main.cpp
    g++ FibLFSR.hpp -Wall -Werror main.cpp -o main.o FibLFSR.o

FibLFSR: FibLFSR.cpp
    g++ FibLFSR.cpp -o FibLFSR.o -Wall -Werror

test: FibLFSR.o test.cpp test.o
    g++ FibLFSR.hpp -Wall -Werror test.cpp -o test.o FibLFSR.o -lboost_unit_test

all: FibLFSR.o test.cpp test.o
    g++ FibLFSR.hpp -Wall -Werror test.cpp -o test.o FibLFSR.o -lboost_unit_test

clean:
    rm main.o FibLFSR.o test.o
```

Ps1a Source Code: main.cpp

```
// John Simonson
// 2/3/20
#include"FibLFSR.hpp"
using namespace std;

int main(int argc, char** argv){
    FibLFSR a("0110001101100001");
    int temp = a.generate(5);
    cout << a << " " << temp << endl;
    return 0;
}
```

Ps1a Source Code: test.cpp

```
// John Simonson
// test.cpp for PS1a


#include <iostream>
```

```
6  #include <string>
7
8  #include"FibLFSR.hpp"
9
10 #define BOOST_TEST_DYN_LINK
11 #define BOOST_TEST_MODULE Main
12 #include <boost/test/unit_test.hpp>
13
14 BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
15
16   FibLFSR l("0110001101100001");
17   BOOST_REQUIRE(l.step() == 1);
18   FibLFSR S("0110001101100001");
19   BOOST_REQUIRE(S.generate(5) == 19);
20 }
```

Ps1a Source Code: FibLFSR.hpp

```
1  // John Simonson
2  // FibLFSR.hpp
3  // 2/3/20
4  #ifndef FIBLFSR_H
5  #define FIBLFSR_H
6  #endif
7  #include<iostream>
8  #include<string>
9  #include<cmath>
10 using namespace std;
11 class FibLFSR {
12 public:
13     FibLFSR(string seed);
14     int step();
15     int generate(int k);
16     friend ostream & operator <<(ostream& out, const FibLFSR c);
17 private:
18     string num;
19 };
```

Ps1a Source Code: FibLFSR.cpp

```
1  // John Simonson
2  // FibLFSR.cpp
```

```
3  // 2/3/20
4  #include"FibLFSR.hpp"
5  using namespace std;
6
7  FibLFSR::FibLFSR(string seed){
8      this->num = seed;
9  }
10
11 ostream & operator <<(ostream& out, const FibLFSR c){
12         out << c.num;
13         return out;
14 }
15
16 int FibLFSR::step(){
17     int temp = this->num[0] ^ this->num[2];
18     temp = temp ^ this->num[3];
19     temp = temp ^ this->num[5];
20     int i;
21     for(i = 0; i <= 14; i++){
22         this->num[i] = this->num[i+1];
23     }
24     this->num[15] = '0' + temp;
25     return temp;
26 }
27
28 int FibLFSR::generate(int k){
29     string output;
30     for(int i = 0; i < k; i++){
31     output += to_string(this->step());
32     }
33     int x = stoi(output, nullptr, 2);
34     return x;
35 }
```