```cpp
#include "LFSR.hpp"

#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(fiveBitsTapAtTwo)
{

  LFSR l("00111", 2);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 0);

  LFSR l2("00111", 2);
  BOOST_REQUIRE(l2.generate(8) == 198);
}
```

```
#ifndef LFSR_HPP
#define LFSR_HPP

#include <iostream>
#include <string>

class LFSR
{

public:
  LFSR(std::string user_seed, int user_tap);
  ~LFSR();
  int step();
  int generate(int k);

  friend std::ostream& operator<< (std::ostream &out, LFSR &lfsr);

private:
  std::string seed;
  int tap;

};


#endif
```

```cpp
#include "LFSR.hpp"

LFSR::LFSR(std::string user_seed, int user_tap)
{
  tap = user_tap;
  seed = user_seed;
}

LFSR::~LFSR()
{

}

int LFSR::step()
{

  int bit;
  char c_bit;

  if (seed.at(0) == seed.at((seed.size()-1) - tap))
    {
      bit = 0;
      c_bit = '0';
    }
  else
    {
      bit = 1;
      c_bit = '1';
    }

  for(unsigned int i = 0; i < (seed.size()-1); i++)
    {
      seed.at(i) = seed.at(i+1);
    }
  seed.at(seed.size()-1) = c_bit;

  return bit;
}

int LFSR::generate(int k)
{
  int gen = 0;

  for (int i =0; i < k; i++)
    {
      gen = gen*2 + LFSR::step();
    }

  return gen;
}

std::ostream& operator<<(std::ostream &out,LFSR &lfsr)
{
  out << lfsr.seed;

  return out;
}
```

```
C=g++ -g -Wall --std=c++98 -Werror
E=.cpp
O=test.o LFSR.o
P=ps2a
BOOST= -lboost_unit_test_framework
#SFML= -lsfml-graphics -lsfml-window -lsfml-system
all: $(P)
$(P):$(O)
	$(C) -o $(P) $(O) $(BOOST)

$(E).o:
	$(C) -c $<  -o $@

clean:
	rm $(O) $(P)
```