COM6014: Practical Task 1. Linear Feedback Shift Register (LFSR) Implementation

# General

LFSRs underpin many stream ciphers, e.g., the ‘classical combining stream   
cipher’ discussed in lectures. For the purposes of this practical two LFSR implementations are of interest:

1. a basic ***specific*** one defined by the figure below.
2. a general one (class) that can be configured in various ways.

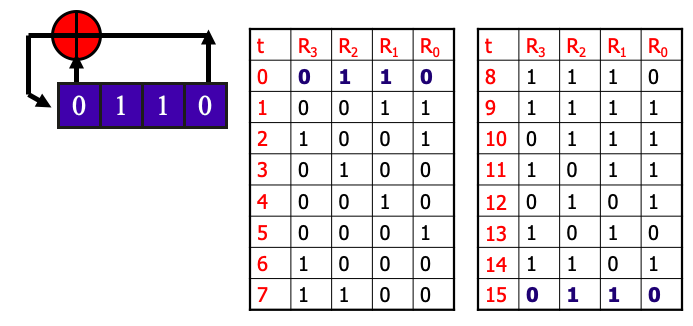
If you are not confident about implementation then do (a) and then do (b). If you are confident, then do (b) and instantiate it to give (a)!

[*In the next practical you will need to implement three different LFRSs. You could use your experience here to simply implement three specific ones or use your general LFSR class and instantiate it appropriately three times.*]

In this practical description I often refer to “by some appropriate method (or similar)” or refer to “class”. This is because all my implementations are in an OO language (Java, in fact). **However, you are free to choose any language you like.**

# The Basic LFSR Implementation

The basic LFSR to implement is one given in lectures:



Your LFSR implementation should:

* allow you to set the initial state via an appropriate method (or similar).
* allow you to get (read) the LFSR state by some appropriate method (or similar).
* allow you to get the *next stream bit* by some appropriate method (or similar). Getting a stream bit should cause the state to update appropriately.

For the basic LFSR I assume the linear feedback function is ‘hardwired’ to be that shown above and is not configurable.

Now, using your implementation, write a program that:

* sets the initial state to 0110
* gets the LFSR state and then gets the next stream bit, and then prints these out to the screen or to a file. **Do this 30 times**.

The output should be the first 15 states repeated twice. Following each read state your next stream bit should be the least significant bit value of that state, i.e., the bit values given in column R0. Run your program to confirm that this is so.

Note that when you have done **15** iterations you should be back at the initial state.

# The General (Reconfigurable) LFSR

Here you should implement an LFSR class (or similar) that provides methods for:

* setting and getting the **size** of the LFSR, i.e. the number of bits in the register. (The basic LFSR implementation above has a size of 4 bits.)
* setting and getting the **initial state** of the LFSR
* setting and getting the ‘**tap sequence**’ used. The tap sequence is the set of indices that define the linear feedback function: the feedback bit value will be the XOR sum of the register values indexed by the tap bits.
  + You can supply the tap sequence as a set of indices of the relevant bits to the appropriate set method. It is also possible to implement the set method so that you supply the index of just one tap bit at a time.
* getting the next stream bit (and altering the state accordingly).
* other methods that help you implement the above.

You’ll need one or more basic constructor methods if you are using an OO language.

You can see that the general idea is to develop software that allows the size of the register and the feedback function to be configurable. **You can assume that the register size is at most 32 bits.**  Write a program that instantiates your general LFSR to the basic version (a) and then carries out the actions indicated in section 2.