```
classdef BitShiftRegister < handle
    properties
        bits = []
        taps = []
        size = 0;
        num_taps = 0;
    end

    methods
        function BSR = BitShiftRegister(size, taps)
            %BitShiftRegister Construct a bit shift register with taps
            %   taps must be a vector
            fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

            %TODO: are the taps legal? Are there at least two?
            BSR.bits = ones(1,size);
            BSR.size = size;
            BSR.taps = taps; % at least two
            BSR.num_taps = length(taps);
        end

        function tap_result = bit_shift_register( BSR )
            %bit_shift_register Compute the tap result

            %TODO: are the taps legal? Are there at least two?
            xor_bits = zeros(1,BSR.num_taps);
            counter = 1;
            for i = BSR.taps
                xor_bits(counter) = BSR.bits(i);
                counter = counter+1;
            end
            tap_result = recurse_xor(xor_bits);
        end

        function code = update( BSR)
            %update Output the code and update the bits in the register
            code = BSR.bits(BSR.size);
            new_bits = [BSR.bit_shift_register() BSR.bits(1:BSR.size-1)];
            BSR.bits = new_bits;
        end
    end
end
```

Published with MATLAB® R2013b