```
function syndromes = get_syndromes(t, R, GF, prnt_flag)
%{
GET_SYNDROMES computes the syndrome polynomial of the recieved
 codeword based off of
 the value of t representing the amount of errors the BCH/R-S code is
 supposed to correct
Inputs:
    t - number of errors the code is supposed to correct
    R - the recieved codeword of length n
    GF - the cell matrix made up of the elements in GF(2^m) whose
    indices are the powers of alpha + 2
Output:
    syndromes - a power form polynomial that holds the evaluation of R
 for a specific
    value a^i in GF(2^m) where i is {1:2*t} (assumes b = 1 or narrow
 sense)
%}

%NOTE: syndrome polynomial represented in power form will always have
 -1
%(inf) in syndromes(1,end), the lowest order coeff, because S(i) goes
 with
%the power of x^i and since i = 1:2*t the first Syndrome value will
%always be S1

%see if need to set default value of prnt_flag
if ~exist('prnt_flag','var')
    prnt_flag = false;
end

%set initial conditions and initialize syndrome vector
num_synd = 2*t;
syndromes = zeros(1, num_synd+1);
syndromes(1, end) = -1;
m = size(GF{1},2);

if(prnt_flag)
    fprintf(" - S(x) is constructed as a polynomial with a max degree
 of %d.\n", num_synd);
    fprintf(" - The coefficients of S(x) are computed by evaluating
 the\n");
    fprintf("   recevied polynomial, ");
    print_poly("R(x)", R, false);
    fprintf("   at 2t consecutive powers of alpha (2t=%d) over GF(2^
%d)[x].\n", num_synd, m);
    fprintf(" - Each evaluated syndrome is the coefficient of the x-
term\n");
    fprintf("   whose degree matches the power of alpha that R(x) was
\n");
    fprintf("   evaluated at.\n");
    fprintf("  1.) Evaluations of S_1 to S_%d are shown below:\n",
 num_synd);
```

```
end

for i = 1:num_synd
    syndromes(end-i) = EvalPolyGF2(R, i, GF);
    if(prnt_flag)
        if(syndromes(end-i) == -1)
            eval_str = "0";
        elseif(syndromes(end-i) == 0)
            eval_str = "1";
        elseif(syndromes(end-i) == 1)
            eval_str = "a";
        else
            eval_str = sprintf("a^%d",syndromes(end-i));
        end
        fprintf("    S_%d = R(a^%d) = %s\n", i, i, eval_str);
    end
end


end
```

Published with MATLAB® R2018b