```
clear;

clc

S = zeros(1, 22); % Initialize the S vector

S(1,1) = 1; % Create the seed by setting the LSB to 1

DATA_OUT = zeros(1, 2^16); % Initialize a DATA_OUT vector to a large size
next_num = 1;

S_initial = S; % Create the initial S vector so we know when we have run for 1 period

found_period = 0;
period = 0;
disp(S)

zero_run_table = zeros(1,24); %Initialize vectors for counting the zeros and ones runs
ones_run_table = zeros(1,24);
zero_k_count = 0;
ones_k_count = 0;
theoretical_prob = 0.5.^(1:24);

for time=1:4.3e6
    ls_bit = S(1,1); % Store the LSB into a variable
    ms_bit = S(1, 22); % Store the MSB into a variable

    S(1, 22) = S(1, 1); % Set the next state of the MSB to the current value of the LSB
    S(1,1:20) = S(1,2:21); % Bit shift the bits from 2 to 21, to 1 to 20
    S(1, 21) = xor(ls_bit, ms_bit); % XOR the LSB and the MSB together and set that to the 21st bit

    DATA_OUT(1,next_num) = ls_bit; % Store the output into DATA_OUT
    next_num = next_num + 1;

    % If the zero k counter is between 1 and 24, and the LSB is 1,
    % increment the value on the table and reset the zero k counter
    if (zero_k_count > 0 && zero_k_count < 25 && ls_bit == 1)
        zero_run_table(zero_k_count) = zero_run_table(zero_k_count) + 1;
        zero_k_count = 0;
    end

    % If the ones k counter is between 1 and 24, and the LSB is 0,
    % increment the value on the table and reset the ones k counter

    if (ones_k_count > 0 && ones_k_count < 25 && ls_bit == 0)
        ones_run_table(ones_k_count) = ones_run_table(ones_k_count) + 1;
        ones_k_count = 0;
    end

    % If the LSB is 0, and the ones k counter is greater than 0, increment
    % the value on the table and reset the counter to start counting zeros
    if (ls_bit == 0)
        if (ones_k_count > 0)
           ones_run_table(ones_k_count) = ones_run_table(ones_k_count) + 1;
        end
```

```
        ones_k_count = 0;
        zero_k_count = zero_k_count + 1;

    % If the LSB is 1, and the zeros k counter is greater than 0, increment
    % the value on the table and reset the counter to start counting ones
    else
        if (zero_k_count > 0)
            zero_run_table(zero_k_count) = zero_run_table(zero_k_count) + 1;
        end
        zero_k_count = 0;
        ones_k_count = ones_k_count + 1;
    end

    fprintf("here is the state-vector at time %g\n", time);
    fprintf("%g, ", S);
    fprintf("\n\n");
    % Check if we have returned the S vector back to the origial state
    if (S == S_initial)
        fprintf("The state at time %g == the initial state; we are done\n",
time);
        found_period = 1;
        period = time;
        break;
    end
end

if (found_period == 1)
    %Printing out final data after the period has been found
    fprintf("\nFound period = %g clock ticks, here are the random bits\n",
period);
    fprintf("%g, ", DATA_OUT(1,1:period));
    fprintf("\n\n");

    fprintf("Here is a decimal representation\n");
    %Finding the number of total bytes in the period of the run
    num_bytes = floor(period/8);

    %Converting the DATA_OUT from an array of 8 bit binary numbers to its
    %decimal representation
    random_numbers = zeros(1, 2^16/8);
    for j=1:num_bytes
        start_index = (j-1)*8+1;
        end_index = start_index+8-1;

        BITS = DATA_OUT(1,start_index:end_index);

        integer = bits2num(BITS);
        random_numbers(1, j) = integer;
        fprintf("%g, ", integer);
    end
    fprintf("\n")
    fid = fopen("my_random_numbers.m", "w");
    fprintf(fid,"%3g ", random_numbers);
    fclose(fid);
else
   fprintf("DID NOT FIND PERIOD! \n");
end

%Creating the table for 0-runs and 1-runs occurences and probability
zeros_cond_prob(1:24) = zero_run_table(1:24)/sum(zero_run_table);
```

```
114 ones_cond_prob(1:24) = ones_run_table(1:24)/sum(ones_run_table);
115
116 zeros_stats = [4,24];
117 zeros_stats(1,1:24) = (1:24);
118 zeros_stats(2,1:24) = zero_run_table;
119 zeros_stats(3,1:24) = zeros_cond_prob;
120 zeros_stats(4,1:24) = theoretical_prob;
121
122 ones_stats = [4,24];
123 ones_stats(1,1:24) = (1:24);
124 ones_stats(2,1:24) = ones_run_table;
125 ones_stats(3,1:24) = ones_cond_prob;
126 ones_stats(4, 1:24) = theoretical_prob;
127
```