Communication Lab II

(ELC3940)

Experiment No.: 06

Object:

Generate (7,4) Hamming Code for 4-bit binary Message signal

G. No: **GL3136**

S. No: **A3EL-02**

F. No: **19ELB056**

Name: **Maha Zakir Khan**

Date of performing experiment: 8|03|2022

Date of report submission: 15|03|2022

Software Used:

MATLAB®, Release 2021a (R2021a), a programming platform designed specifically to analyze and design systems and products. The heart of MATLAB is the MATLAB language, a matrix-based language, it provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.

Procedure:

Note: '+' operator used in the context indicates Boolean OR

'X' operator indicates Matrix Boolean Multiplication

<u>Step 1:</u> Find Generator matrix G:

$$G_{k\text{-by-}n} = [I_k \; P] \text{ or } [P \; I_k]$$

Where:

k = Message Signal Bits

n = Codeword Bits

<u>Step 2:</u> Find Parity-Check matrix (H):

$$H_{(n-k)\text{-by-}n} = [-P' \; I_{n-k}] \text{ or } [I_{n-k} \; -P']$$

Where:

k = Message Signal Bits

n = Codeword Bits

Step 3: Generate valid Codeword (Cm):

$$Cm_{1xn} = m_{1xk} \ X \ G_{kxn}$$

Where m is the message signal

Step 4: Receive Codeword (r) with Error:

$$r_{1 \times n} = Cm + e$$

Where e is the n-bit error

Step 5: Find Syndrome:

$$\text{Syndrome}_{1 \times (n-k)} = r \ X \ H$$

Step 4: Locate Syndrome in the Syndrome Table:

Find the index (j) of the Syndrome in the Syndrome Table and the corresponding error pattern in E

Syndrome Table (S_t) is given by:

$$S_t = R \ X \ H$$

Where: S_t is 2^{n-k} by n binary matrix

R is given as:

$$R_i = C_r + E_i$$

Where:

C_r is any valid codeword

E is all possible combination of 1-bit Error, given as a 7x7 matrix for n = 7 indexed by i

Step 4: Correct error in the Received Codeword (r):

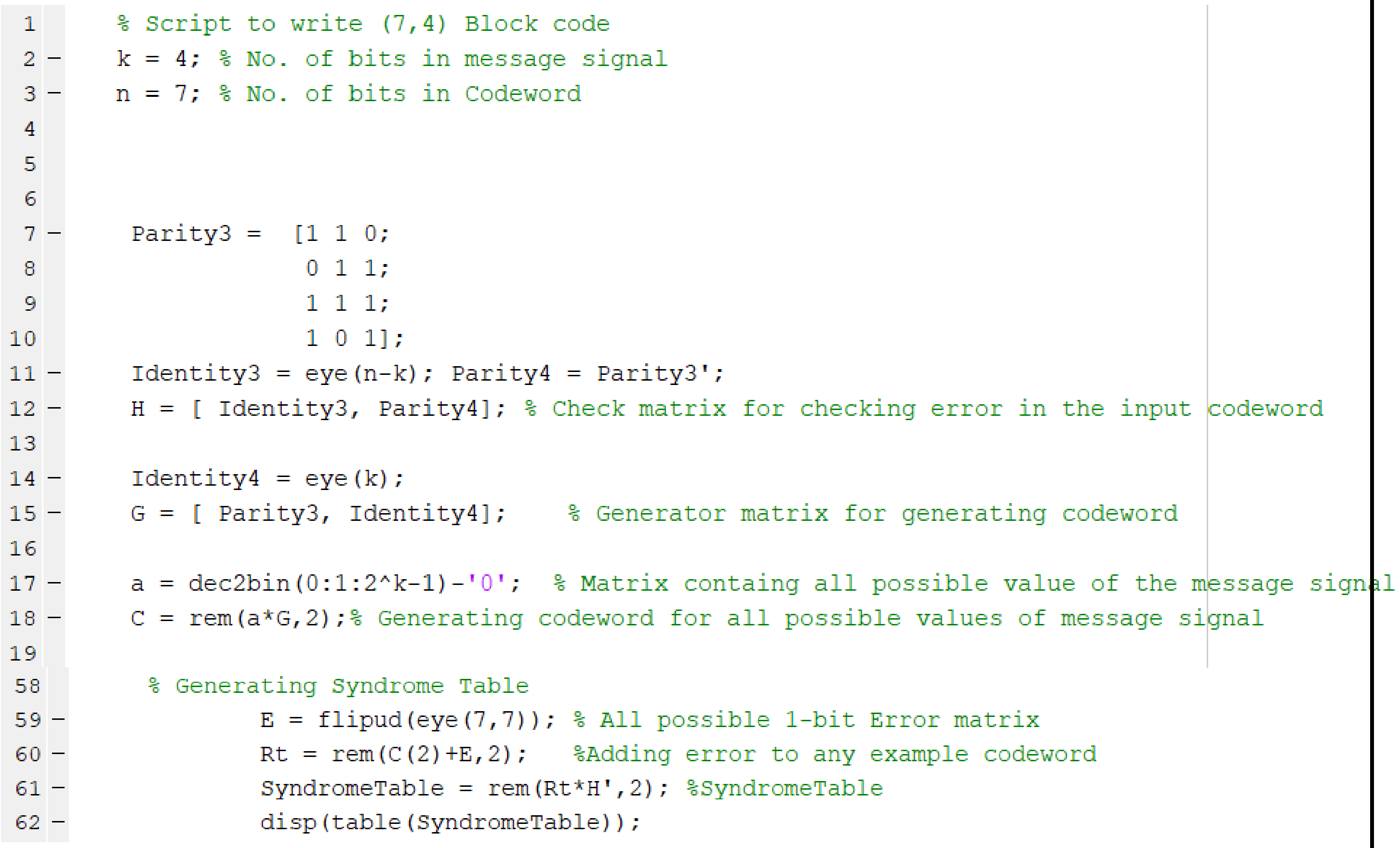
$$C_{Corrected} = r + e$$

Where: e is the detected error pattern given as E_j

$C_{Corrected}$ is the corrected codeword

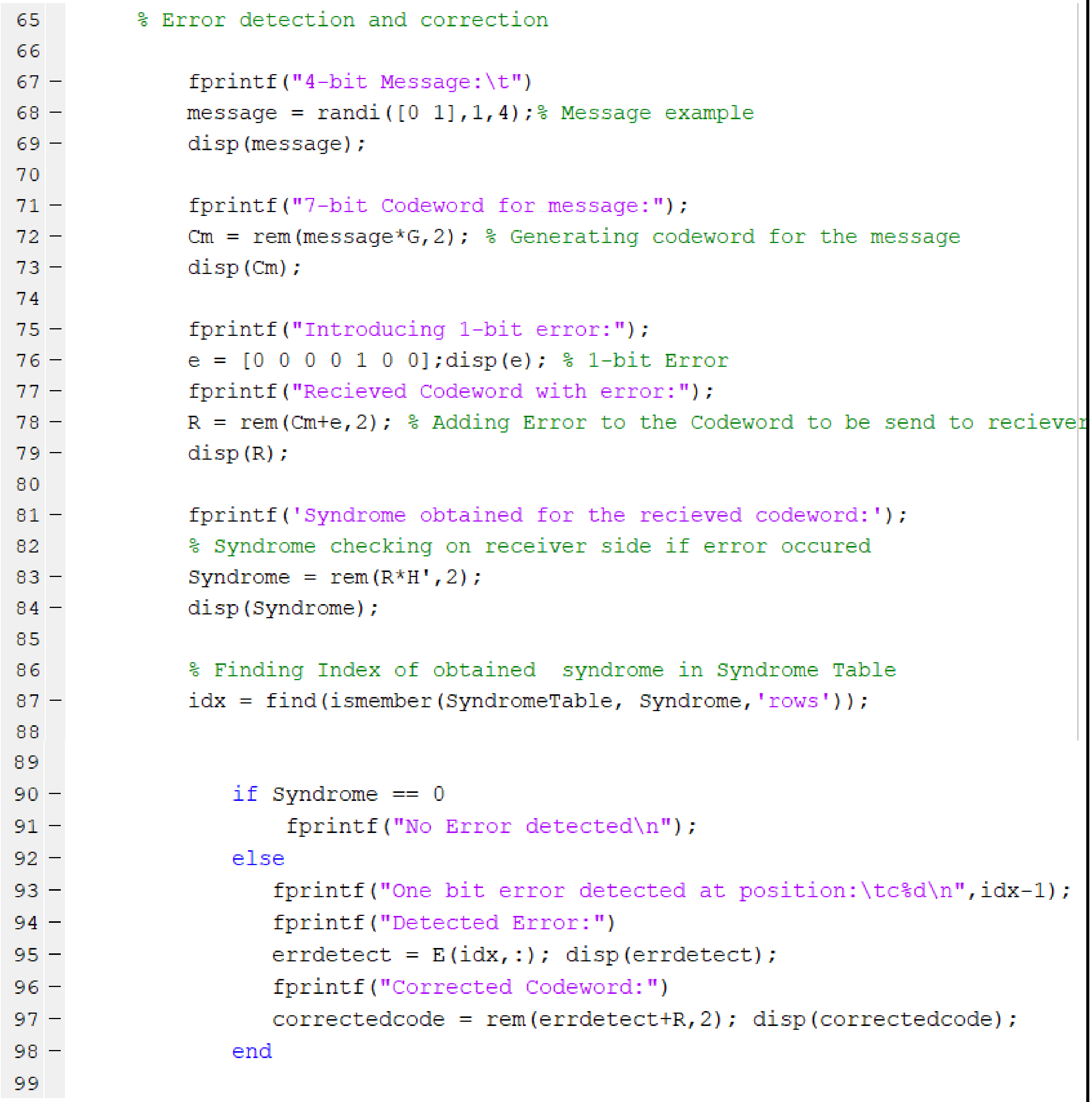
R is the error codeword

Program:



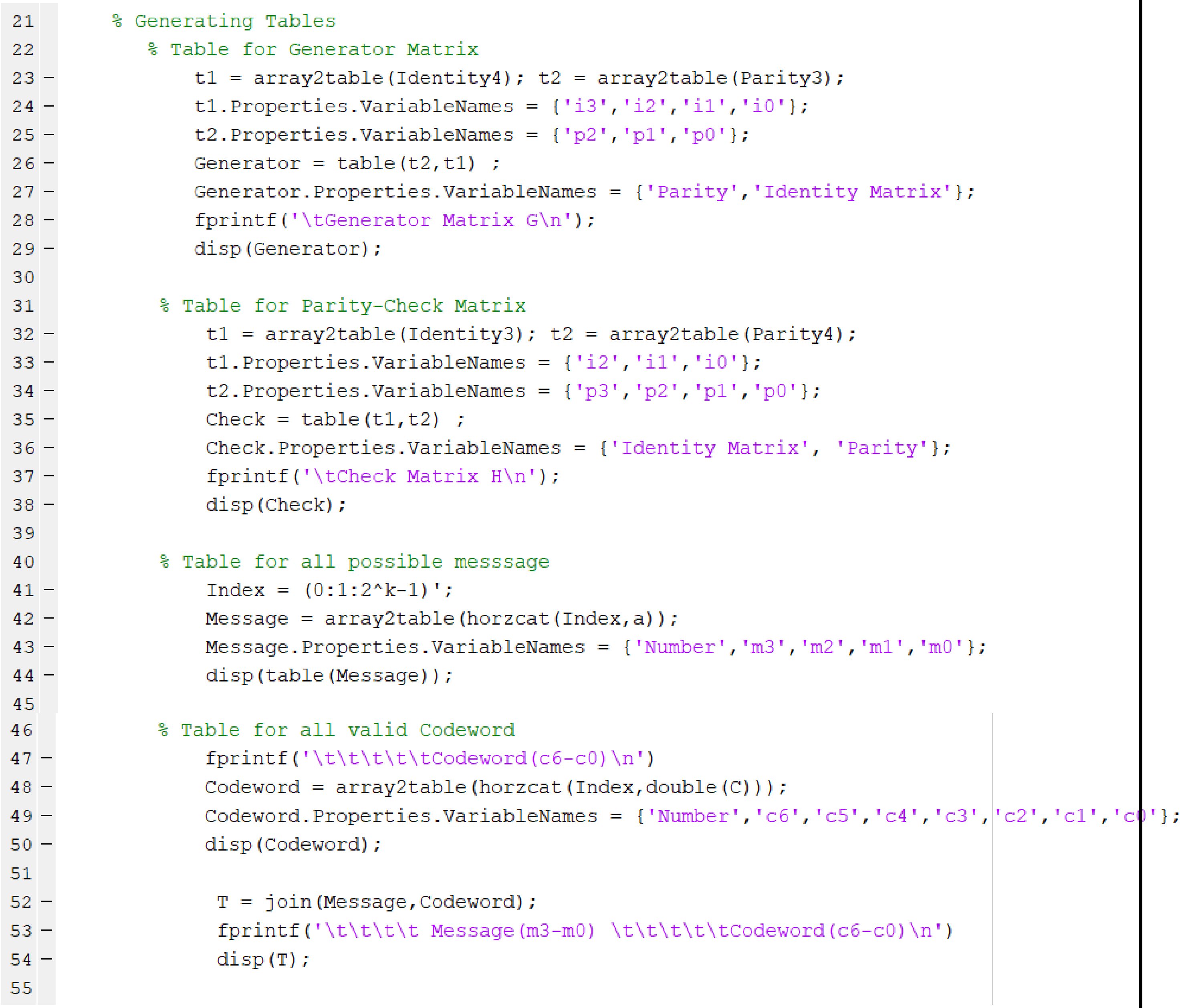
```matlab
% Script to write (7,4) Block code
k = 4; % No. of bits in message signal
n = 7; % No. of bits in Codeword



 Parity3 =  [1 1 0;
             0 1 1;
             1 1 1;
             1 0 1];
 Identity3 = eye(n-k); Parity4 = Parity3';
 H = [ Identity3, Parity4]; % Check matrix for checking error in the input codeword

 Identity4 = eye(k);
 G = [ Parity3, Identity4];    % Generator matrix for generating codeword

 a = dec2bin(0:1:2^k-1)-'0';  % Matrix containg all possible value of the message signal
 C = rem(a*G,2);% Generating codeword for all possible values of message signal

  % Generating Syndrome Table
          E = flipud(eye(7,7)); % All possible 1-bit Error matrix
          Rt = rem(C(2)+E,2);   %Adding error to any example codeword
          SyndromeTable = rem(Rt*H',2); %SyndromeTable
          disp(table(SyndromeTable));
```

Error detection and correction:



```
    % Error detection and correction

        fprintf("4-bit Message:\t")
        message = randi([0 1],1,4);% Message example
        disp(message);

        fprintf("7-bit Codeword for message:");
        Cm = rem(message*G,2); % Generating codeword for the message
        disp(Cm);

        fprintf("Introducing 1-bit error:");
        e = [0 0 0 0 1 0 0];disp(e); % 1-bit Error
        fprintf("Recieved Codeword with error:");
        R = rem(Cm+e,2); % Adding Error to the Codeword to be send to reciever
        disp(R);

        fprintf('Syndrome obtained for the recieved codeword:');
        % Syndrome checking on receiver side if error occured
        Syndrome = rem(R*H',2);
        disp(Syndrome);

        % Finding Index of obtained  syndrome in Syndrome Table
        idx = find(ismember(SyndromeTable, Syndrome,'rows'));


            if Syndrome == 0
                fprintf("No Error detected\n");
            else
               fprintf("One bit error detected at position:\tc%d\n",idx-1);
               fprintf("Detected Error:")
               errdetect = E(idx,:); disp(errdetect);
               fprintf("Corrected Codeword:")
               correctedcode = rem(errdetect+R,2); disp(correctedcode);
            end

```

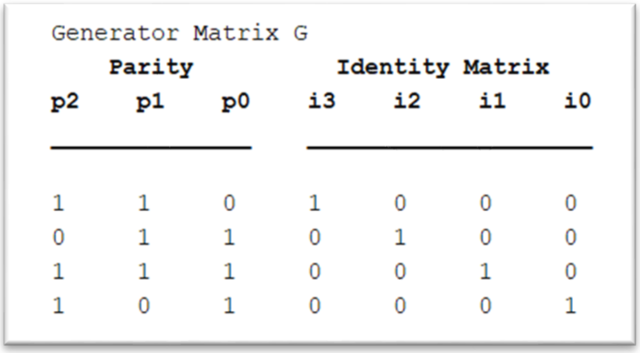
Tables:



```
% Generating Tables
   % Table for Generator Matrix
       t1 = array2table(Identity4); t2 = array2table(Parity3);
       t1.Properties.VariableNames = {'i3','i2','i1','i0'};
       t2.Properties.VariableNames = {'p2','p1','p0'};
       Generator = table(t2,t1) ;
       Generator.Properties.VariableNames = {'Parity','Identity Matrix'};
       fprintf('\tGenerator Matrix G\n');
       disp(Generator);

    % Table for Parity-Check Matrix
        t1 = array2table(Identity3); t2 = array2table(Parity4);
        t1.Properties.VariableNames = {'i2','i1','i0'};
        t2.Properties.VariableNames = {'p3','p2','p1','p0'};
        Check = table(t1,t2) ;
        Check.Properties.VariableNames = {'Identity Matrix', 'Parity'};
        fprintf('\tCheck Matrix H\n');
        disp(Check);

    % Table for all possible messsage
        Index = (0:1:2^k-1)';
        Message = array2table(horzcat(Index,a));
        Message.Properties.VariableNames = {'Number','m3','m2','m1','m0'};
        disp(table(Message));

    % Table for all valid Codeword
        fprintf('\t\t\t\t\tCodeword(c6-c0)\n')
        Codeword = array2table(horzcat(Index,double(C)));
        Codeword.Properties.VariableNames = {'Number','c6','c5','c4','c3','c2','c1','c0'};
        disp(Codeword);

         T = join(Message,Codeword);
         fprintf('\t\t\t\t Message(m3-m0) \t\t\t\tCodeword(c6-c0)\n')
         disp(T);

```

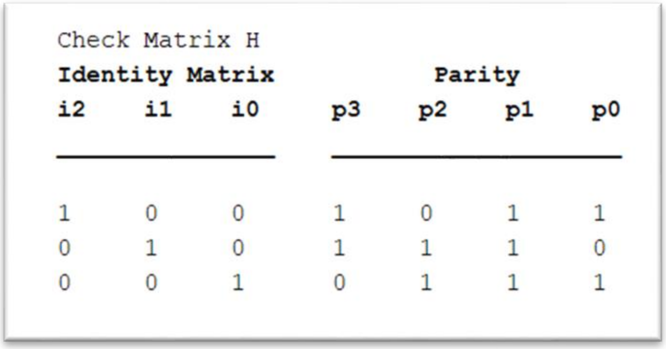
Observation:

Tables:



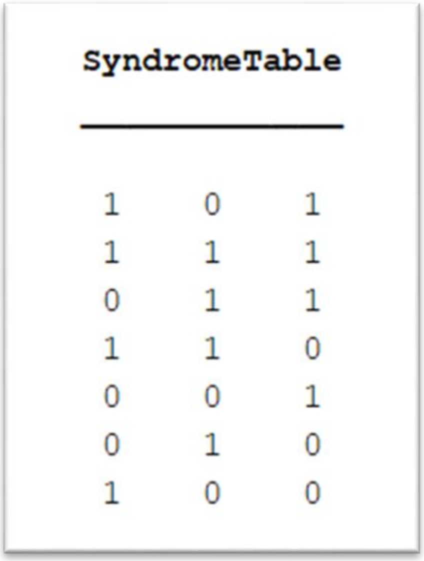
Generator Matrix G

Parity			Identity Matrix			
p2	**p1**	**p0**	**i3**	**i2**	**i1**	**i0**
1	1	0	1	0	0	0
0	1	1	0	1	0	0
1	1	1	0	0	1	0
1	0	1	0	0	0	1

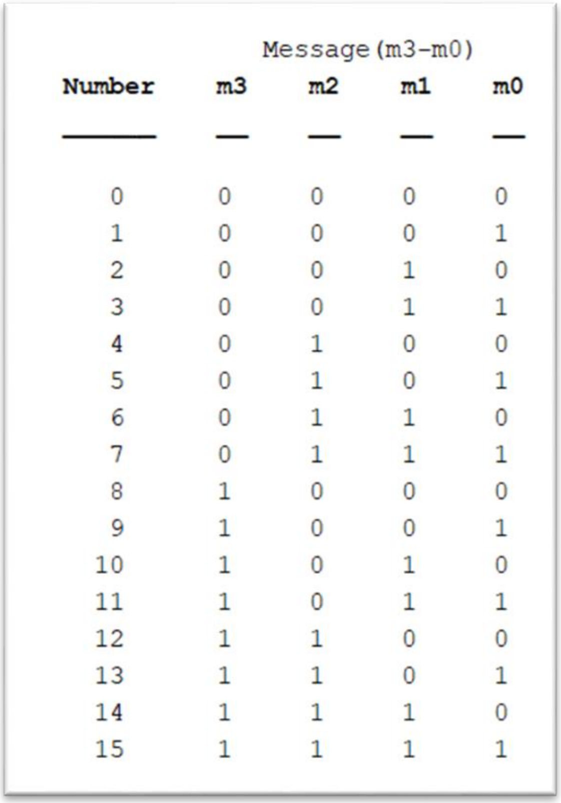


Check Matrix H

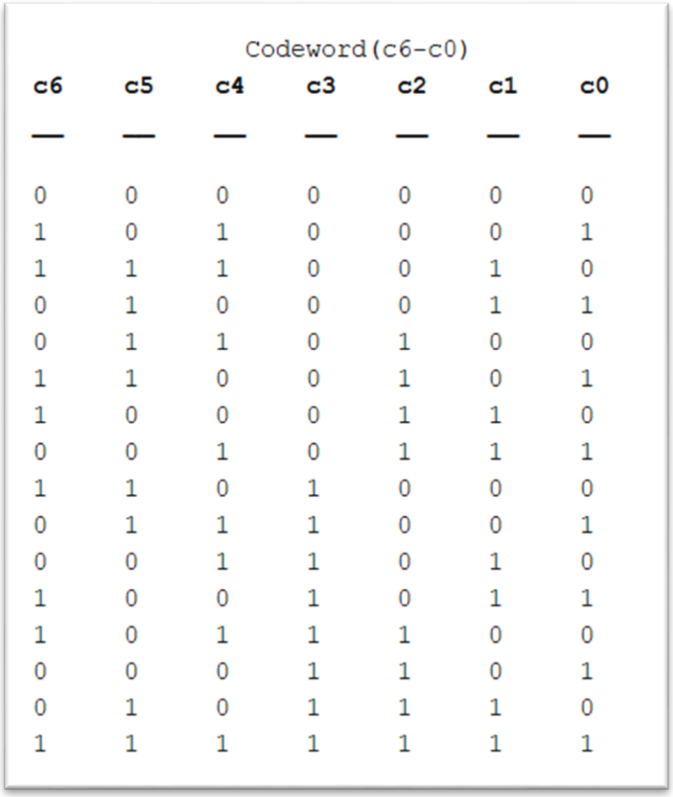
Identity Matrix			Parity			
i2	**i1**	**i0**	**p3**	**p2**	**p1**	**p0**
1	0	0	1	0	1	1
0	1	0	1	1	1	0
0	0	1	0	1	1	1



SyndromeTable		
1	0	1
1	1	1
0	1	1
1	1	0
0	0	1
0	1	0
1	0	0



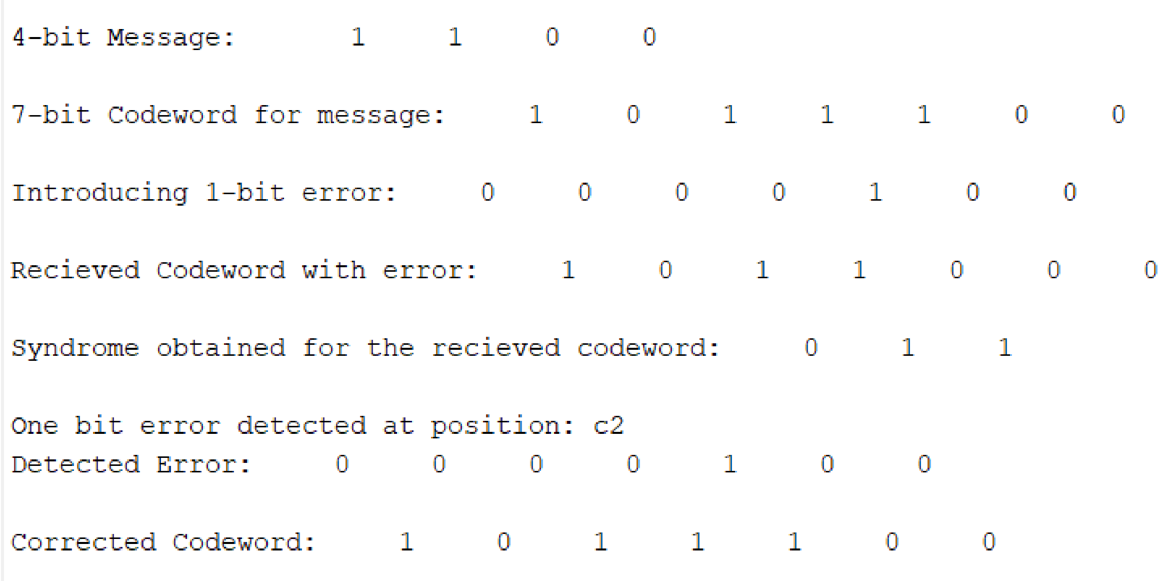
	Message (m3-m0)			
Number	**m3**	**m2**	**m1**	**m0**
______	__	__	__	__
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



			Codeword (c6-c0)			
c6	**c5**	**c4**	**c3**	**c2**	**c1**	**c0**
__	__	__	__	__	__	__
0	0	0	0	0	0	0
1	0	1	0	0	0	1
1	1	1	0	0	1	0
0	1	0	0	0	1	1
0	1	1	0	1	0	0
1	1	0	0	1	0	1
1	0	0	0	1	1	0
0	0	1	0	1	1	1
1	1	0	1	0	0	0
0	1	1	1	0	0	1
0	0	1	1	0	1	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
0	0	0	1	1	0	1
0	1	0	1	1	1	0
1	1	1	1	1	1	1

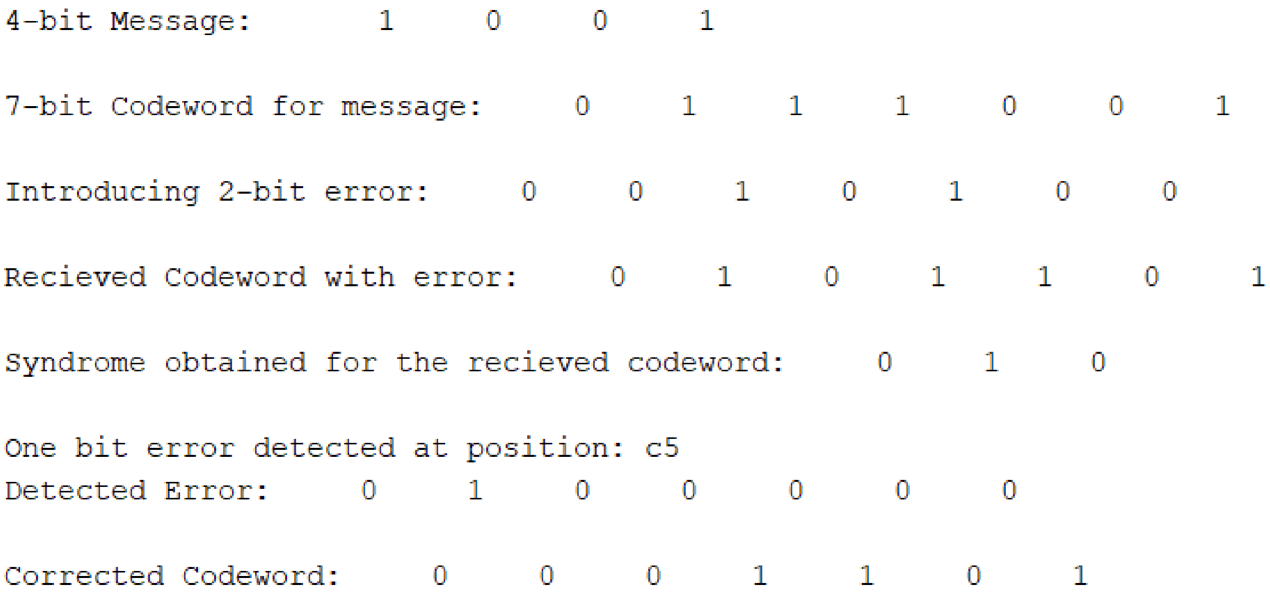
Result:

i.) For 1-bit Error:



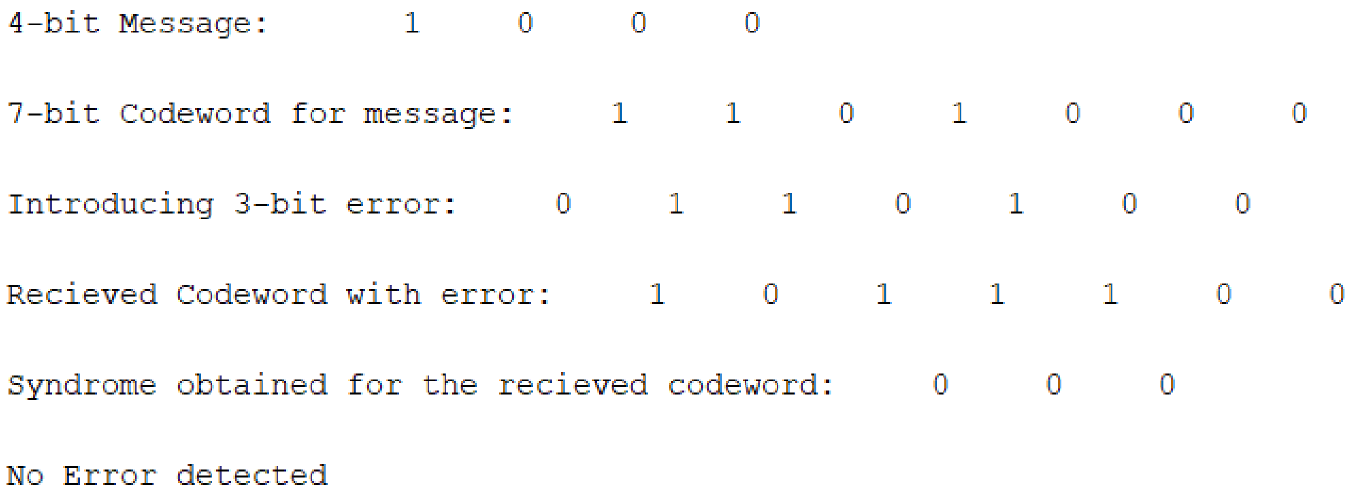
```
4-bit Message:       1     1     0     0

7-bit Codeword for message:     1     0     1     1     1     0     0

Introducing 1-bit error:     0     0     0     0     1     0     0

Recieved Codeword with error:     1     0     1     1     0     0     0

Syndrome obtained for the recieved codeword:     0     1     1

One bit error detected at position: c2
Detected Error:     0     0     0     0     1     0     0

Corrected Codeword:     1     0     1     1     1     0     0
```

ii.) For 2-bit Error:



```
4-bit Message:       1     0     0     1

7-bit Codeword for message:     0     1     1     1     0     0     1

Introducing 2-bit error:     0     0     1     0     1     0     0

Recieved Codeword with error:     0     1     0     1     1     0     1

Syndrome obtained for the recieved codeword:     0     1     0

One bit error detected at position: c5
Detected Error:     0     1     0     0     0     0     0

Corrected Codeword:     0     0     0     1     1     0     1
```

iii.) For 3-bit Error:



```
4-bit Message:        1     0     0     0

7-bit Codeword for message:     1     1     0     1     0     0     0

Introducing 3-bit error:     0     1     1     0     1     0     0

Recieved Codeword with error:     1     0     1     1     1     0     0

Syndrome obtained for the recieved codeword:     0     0     0

No Error detected
```

Discussion:

In this Experiment, I used randomly generated message signal and introduced 1-bit, 2-bit, 3-bit error. It was observed that the code can detect and correct 1-bit error accurately, whereas it detected but failed to correct 2-bit error, and failed to detect 3 or more-bit errors as expected:

$n= 7, k = 4$

Hamming distance $d_{min} = n-k = 7-4 = 3$

Error bits that can be detected = $t_d = d_{min} - 1 = 2$

Error bits that can be corrected = $t_c = [(d_{min} -1)/2] = [(3-1)/2] = 1$