## Exp9.m

```
1 clc;
 2 clear all;
 4 % Input generator matrix
 5 g = input('Enter the generator matrix: ');
 6 disp('G =');
 7 disp(g);
 8
 9 % Determine the order of the linear block code
10 [n, k] = size(transpose(g));
11 disp('The order of the linear block code for given generator matrix is: ');
12 disp(['n = ', num2str(n), ', k = ', num2str(k)]);
13
14 % Generate all possible codewords
15 u = zeros(2^k, k);
16 for i = 1:2^k
17
       for j = k:-1:1
18
           if rem(i - 1, 2^{(-j + k + 1)}) >= 2^{(-j + k)}
19
               u(i, j) = 1;
20
           else
21
               u(i, j) = 0;
           end
22
23
       end
24 end
25
26\ \% Calculate codewords by multiplying message vectors with G
27 c = rem(u * g, 2);
28 disp('The possible codewords are: ');
29 disp(c);
30
31 % Calculate minimum Hamming distance
32 disp('The minimum Hamming distance dmin for given block code = ');
33 d_{min} = min(sum(c(2:2^k, :)')); % Exclude the all-zero codeword
34 disp(d_min);
35
36 % Input received codeword
37 r = input('Enter the received code word: ');
38
39 % Calculate the parity-check matrix H
40 p = g(:, n - k + 1:n);
41 h = [transpose(p), eye(n - k)];
42 disp('Parity-check matrix H: ');
43 disp(h);
44
45 % Calculate syndrome of received codeword
46 ht = transpose(h);
47 s = rem(r * ht, 2);
48 disp('Syndrome of the given codeword is: ');
49 disp(s);
50
51 % Error correction based on syndrome
52 error_pos = -1;
53 for i = 1:n
54
       if isequal(ht(i, :), s)
55
           error_pos = i;
56
           r(i) = 1 - r(i); % Flip the bit at the error position
57
           break;
58
       end
59 end
60
61 % Display results
62 if error_pos > 0
       disp(['The error is in bit: ', num2str(error_pos)]);
63
64 else
65
       disp('No single-bit error detected');
66 end
67
```

about:blank 1/2

11/12/24, 10:08 AM about:blank

```
68 disp('The corrected codeword is: ');
69 disp(r);
70
```

Printed for: swarajnangare36@gmail.com Powered by Octave Online http://octave-online.net

about:blank 2/2