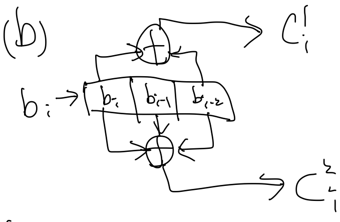
1.

(a) $k=1 \quad n=2 \quad R=\frac{1}{2}$ #

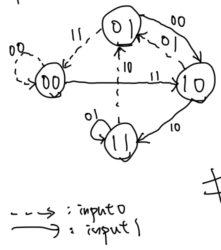
(b)



#

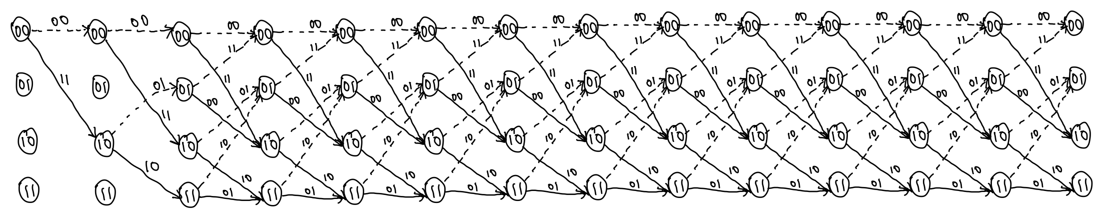
(c)

b_i	b_{i-1}	b_{i-2}	C_i^1	C_i^2	b_{i-1}	b_{i-2}^{new}
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	1	1	1



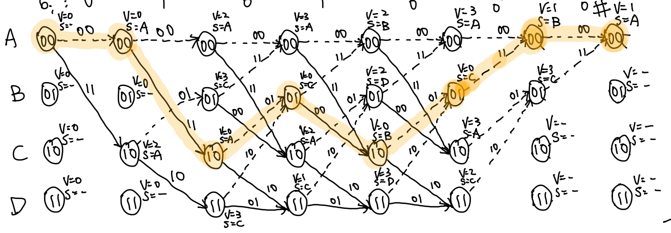
#

(d)



(e)

y :	00	11	01	00	01	01	00
$C_i^1 C_i^2$:	00	11	01	00	01	11	00
b_i :	0	1	0	1	0	0	0



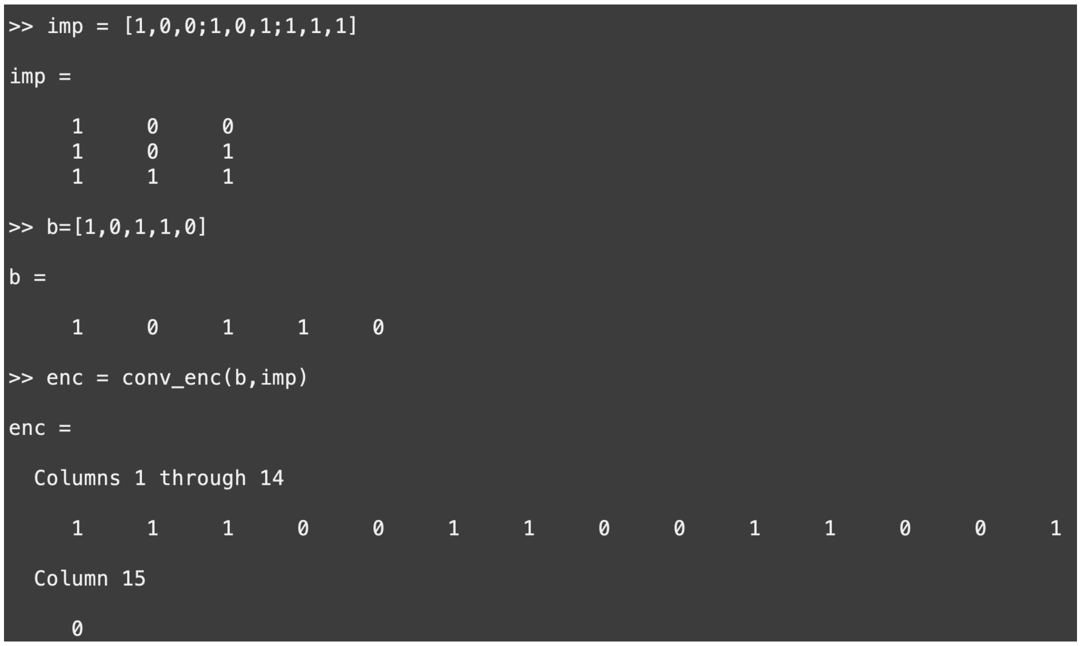
#

(f)

$d_{H\,min} = 1$ #

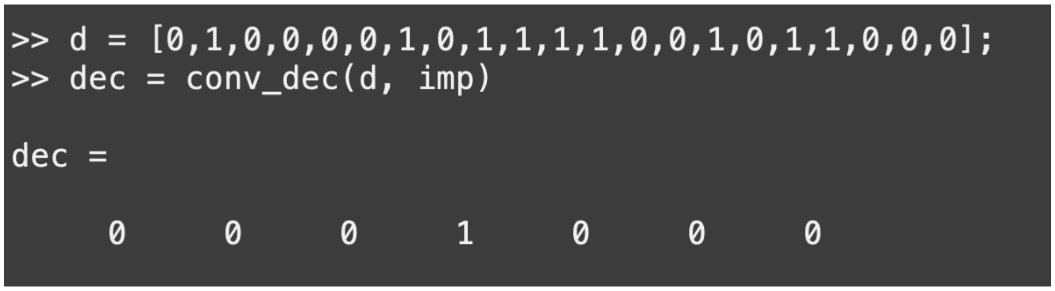
B10901184 錡亭勳 Lab5

2.
(a).

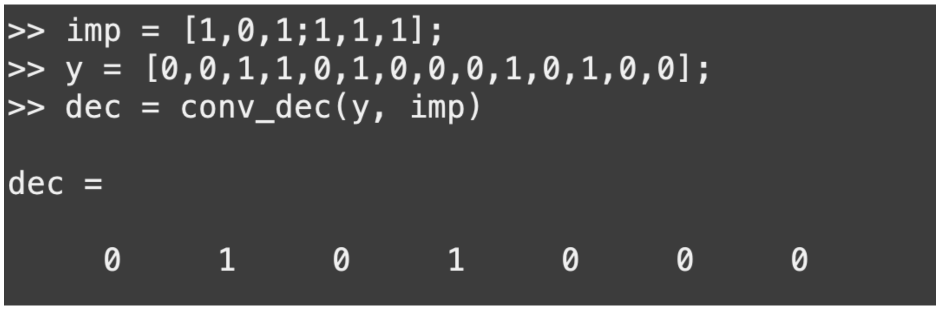


```
>> imp = [1,0,0;1,0,1;1,1,1]

imp =

     1     0     0
     1     0     1
     1     1     1

>> b=[1,0,1,1,0]

b =

     1     0     1     1     0

>> enc = conv_enc(b,imp)

enc =

  Columns 1 through 14

     1     1     1     0     0     1     1     0     0     1     1     0     0     1

  Column 15

     0
```

(b).



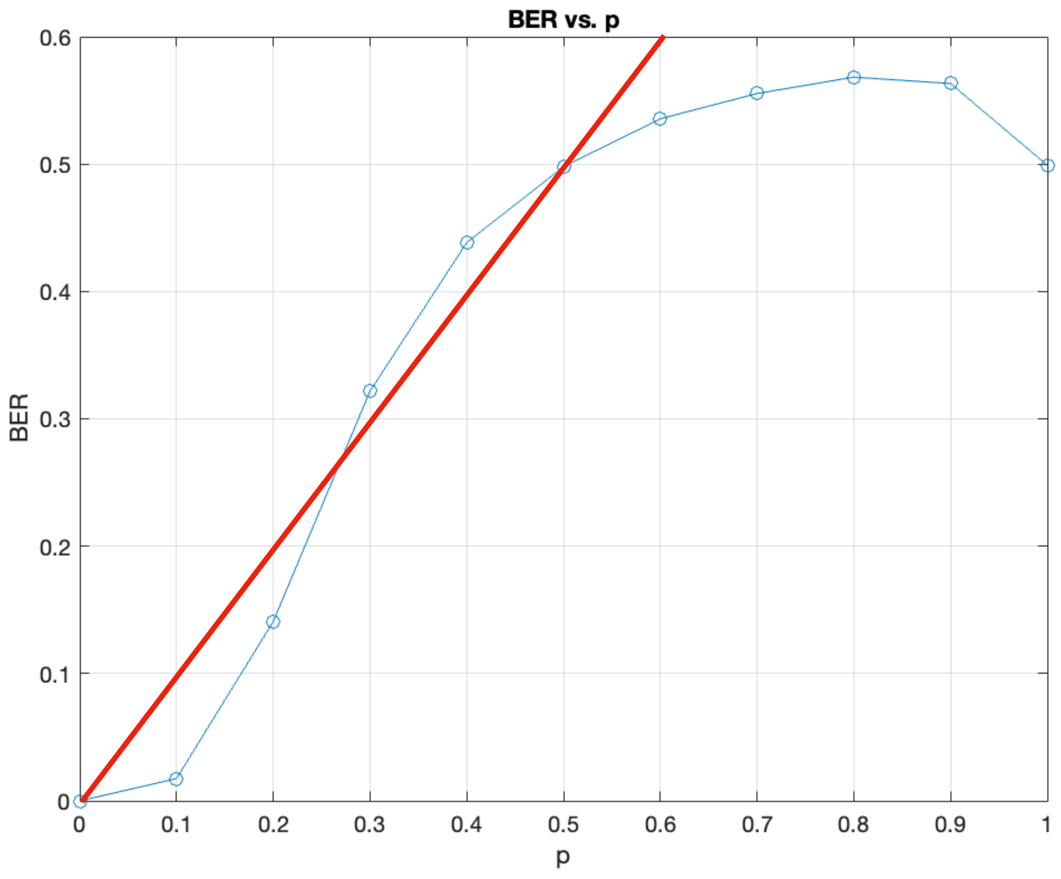
```
>> d = [0,1,0,0,0,0,1,0,1,1,1,1,0,0,1,0,1,1,0,0,0];
>> dec = conv_dec(d, imp)

dec =

     0     0     0     1     0     0     0
```



```
>> imp = [1,0,1;1,1,1];
>> y = [0,0,1,1,0,1,0,0,0,1,0,1,0,0];
>> dec = conv_dec(y, imp)

dec =

     0     1     0     1     0     0     0
```

Same as Problem 1 (e).

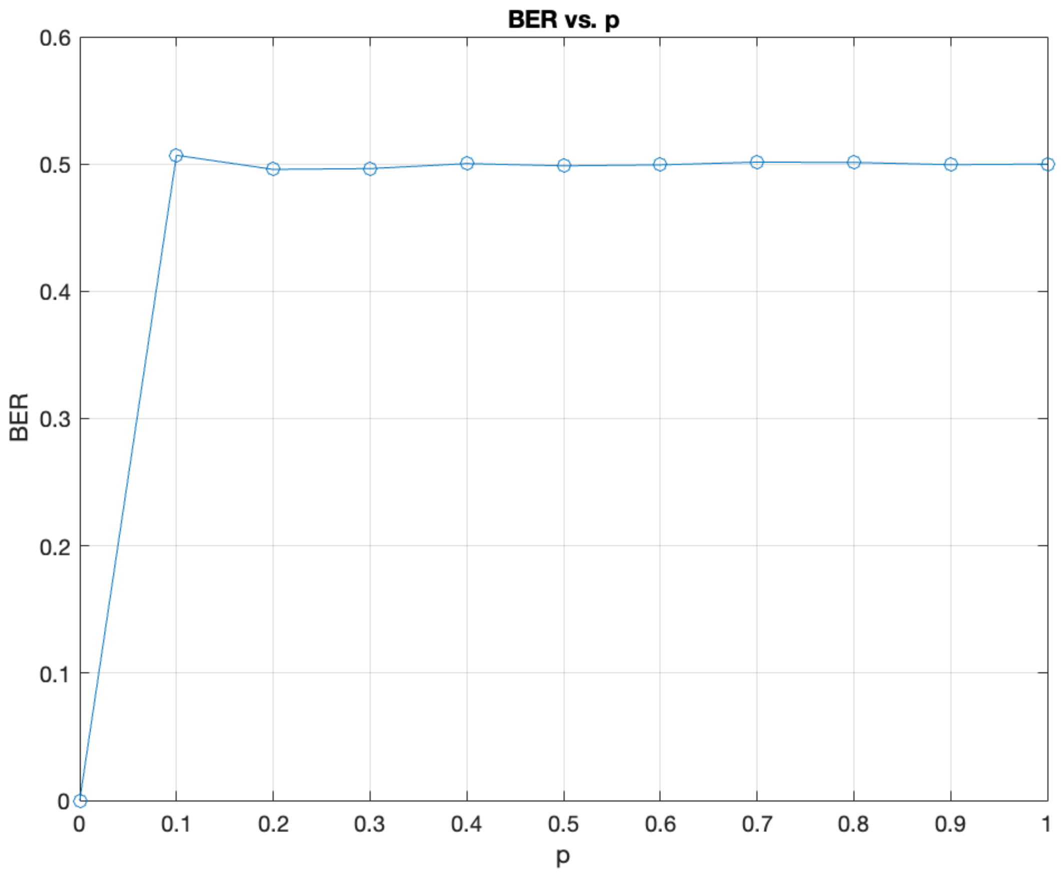
(c).



We can see that only when p<=0.2, the convolution code can fix the bit flip error. When p > 0.2, the error is the same as doing nothing (without encoding, the red line shown in the figure). This tells us that this convolution code can work only for small errors but not big errors.

When p>1/2, the decoded bits' BER approaches constant 0.5, since when an error occurs more frequently than a correct bit, all it can do is randomly guess the bit strings.

3.



We can see that the channel coding really does nothing, even worsen the BER. This is because, with only two generators, the encoded bit strings are not capable of correcting even BER=0.1. Compared to Problem 2, we can see that we need more generators to have a more fault-tolerant channel coding.