In [1]:

```
import numpy as np
G = np.array([1,0,1,1])
M = np.array([1,0,0,1])
print(G)
print(M)
```

```
[1 0 1 1]
[1 0 0 1]
```

In [2]:

```
g = np.poly1d(G)
print(g)
m = np.poly1d(M)
print(m)
```

```
   3
1 x + 1 x + 1
   3
1 x + 1
```

In [3]:

```
#n-k shift r
r = g.order
parity = np.zeros((r+1))
parity[0] = 1
parity = np.poly1d(parity)
print(parity)
```

```
   3
1 x
```

In [4]:

```
m_shifted = np.polymul(m,parity)
print(m_shifted)
```

```
   6     3
1 x + 1 x
```

In [5]:

```
#dividing shifted by g
q,rem = np.polydiv(m_shifted,g)
print(rem)
```

```
   2
1 x + 1 x
```

In [6]:

```
R = rem.c
R = R % 2
print(R)
```

```
[1. 1. 0.]
```

In [7]:

```
C = np.hstack((M,R))
C = C.astype(int)
print(C)
```

```
[1 0 0 1 1 1 0]
```

DECODING

In [8]:

```
#introducing error at 3rd bit
```

```
RC = C
RC[3] = int(not RC[3])
print(RC)
```

```
[1 0 0 0 1 1 0]
```

In [9]:

```
rc = np.poly1d(RC)
print(rc)
```

```
 6  2
1 x + 1 x + 1 x
```

In [10]:

```
q, s = np.polydiv(rc,g)
print(s)
```

```
 2
2 x + 3 x + 1
```

In [11]:

```
#Syndrome
S = s.c
S = S % 2
print(S)
```

```
[0. 1. 1.]
```

In [12]:

```
#Checking for errors
if S.all == 0:
  print("No Error")
else:
  print("Error")
```

```
Error
```