EX.NO.6.

Hamming code:-

1b

Receiver.py

```
def calc_parity_positions(m):
    r=0
    while 2**r < m+r+1:
        r+=1
    return r
def read_form_channel():
    with open("channel.txt","r") as f:
     return f.read()
def detect_and_correct_error(data, r):
    n = len(data)
    result = list(data)
    error_pos = 0

 for i in range(r):
    idx = 2**i -1
    parity = 0
    for j in range(1,n+1):
        if j & (2**i) != 0:
            parity ^= int(result[j-1])
   error_pos += parity * (2**i)

 if error_pos >0:
    print(f"ERROR detection at position:
            {error_pos}")
    result[error_pos-1] = '1' if result
            [error_pos-1] == '0' else '0'
    print(f"Corrected data: {''.join
            (result)}")
```

```
else:
    print ("No errors detected.")
return ''. join(result)

def remove_parity_bits (data, r):
    n = len (data)
    result = []
    for i in range (1, n+1):
     if i & (i-1) == 0:
      continue.
    result. append (data [i-1])
return ''. join(result)

def binary_to_text (binary):
    text = ''. join([chr (int (binary
[i: i+8], 2)) for i in range
(0, len (binary), 8)])
return text

def hamming_decode ():
   encoded_data = read_from_Channel ()
   m= len(encoded_data)
   r= calc_parity_position (m-len(
   [i for i in range(m) if i &(i+1)
   == 0]))
 corrected_data = detect_and_correct
 _error (encoded_data , r)
 data_without_parity = remove_parity_
 bits( corrected_data, r)
 decoded_text = binary to_text
 (data_without_parity)
```

```
return decoded_text.

decoded_text = hamming_decode()
print(f" Decoded text: {Decoded_text}")
```

SENDER.PY

```
def text_to_binary (text):
    return ''. join(format(ord(i), '08b'
for i in text)

def calc_parity_positions(m):
    r=0
    while 2**r < m+r+1 :
        r+=1
    return r

def insert_parity_bits(data, r):
    n= len(data)
result = ['0']*(n+r)
j=0
for i in range(1, len(result)+1):
  if i& (i-1)==0:
    continue
result[i-1] = data(j)
  j+=1
return ''.join(result)

def set_parity_bits(data, r);
    n= len(data)
    result = list(data)
```

```
for i in range (r):
  idx = 2**i - 1
  parity = 0
for j in range (1, n+1):
  if j & (2**i) != 0:
parity ^= int (result [j-1])
result [idx] = str (parity)
return ''.join(result)

def hamming_encode (text):
 binary_data = text_to_binary (text)
m = len (binary_data)
r = calc_parity_positions (m)
data_with_parity = insert_parity_bits
   (binary_data, r)
encoded_data = set_parity_bits(data_
with_parity, r)
return encoded_data

def save_to_channel (encoded_data):
 with open ("channel.txt", "w") as f:
  f.write (encoded_data)

text = input ("Entr text to send: ")
encoded_data = hamming_encode(text)
save_to_channel (encoded_data)
print (f"encoded data saved to
"channel.txt": {encoded_data}")
```

channel.txt

110 111001000001110010101 0011 000 1010
11000 1101111 0010000000 1101001000 1
0000000 11 0000 101101101001 0000000 111
0011 0000 1011 000 100 10000 100111 0011
001101001 0111 0011011010000.

9/10