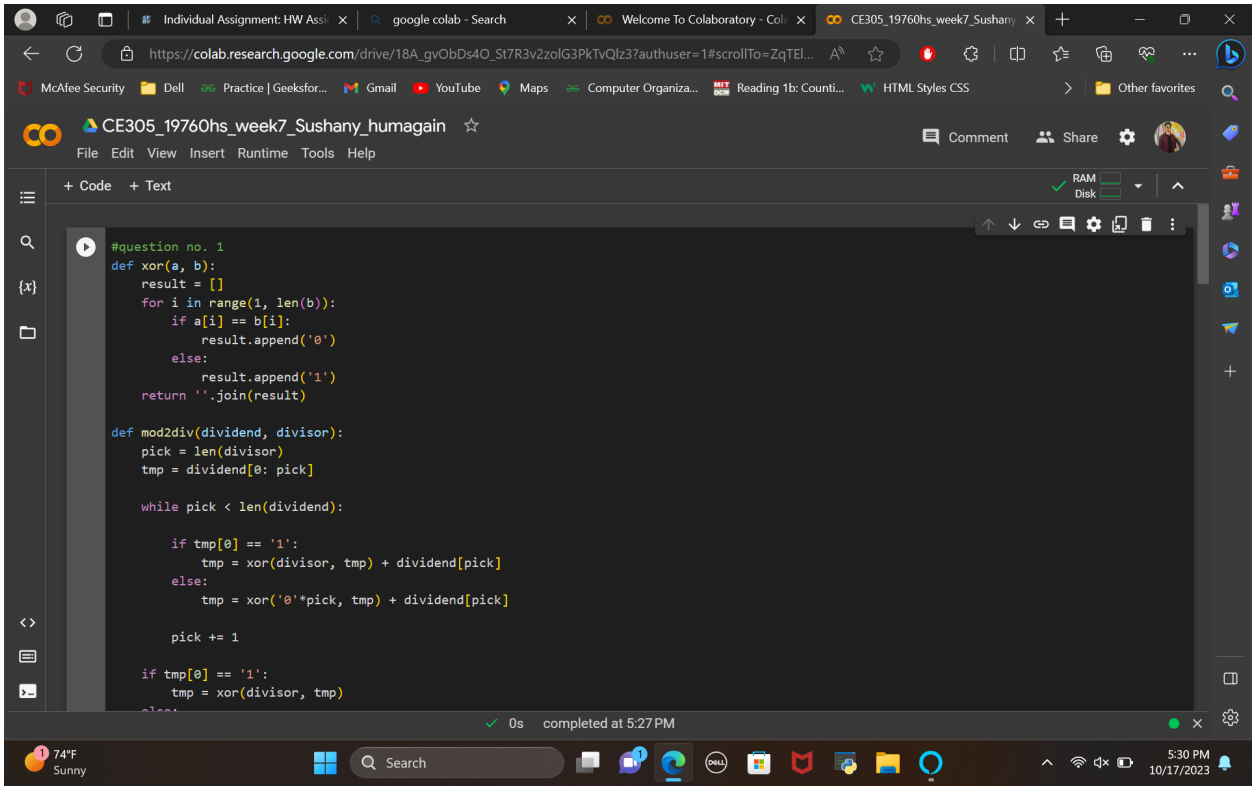
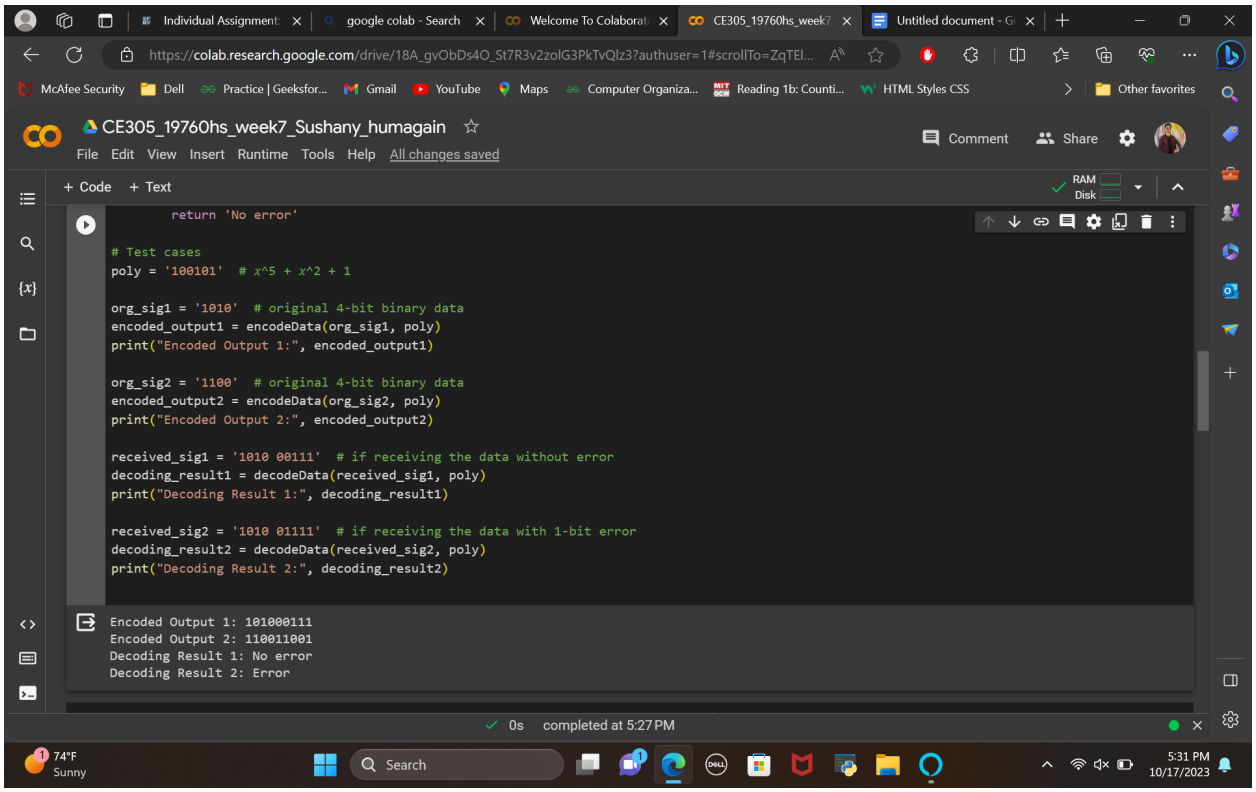
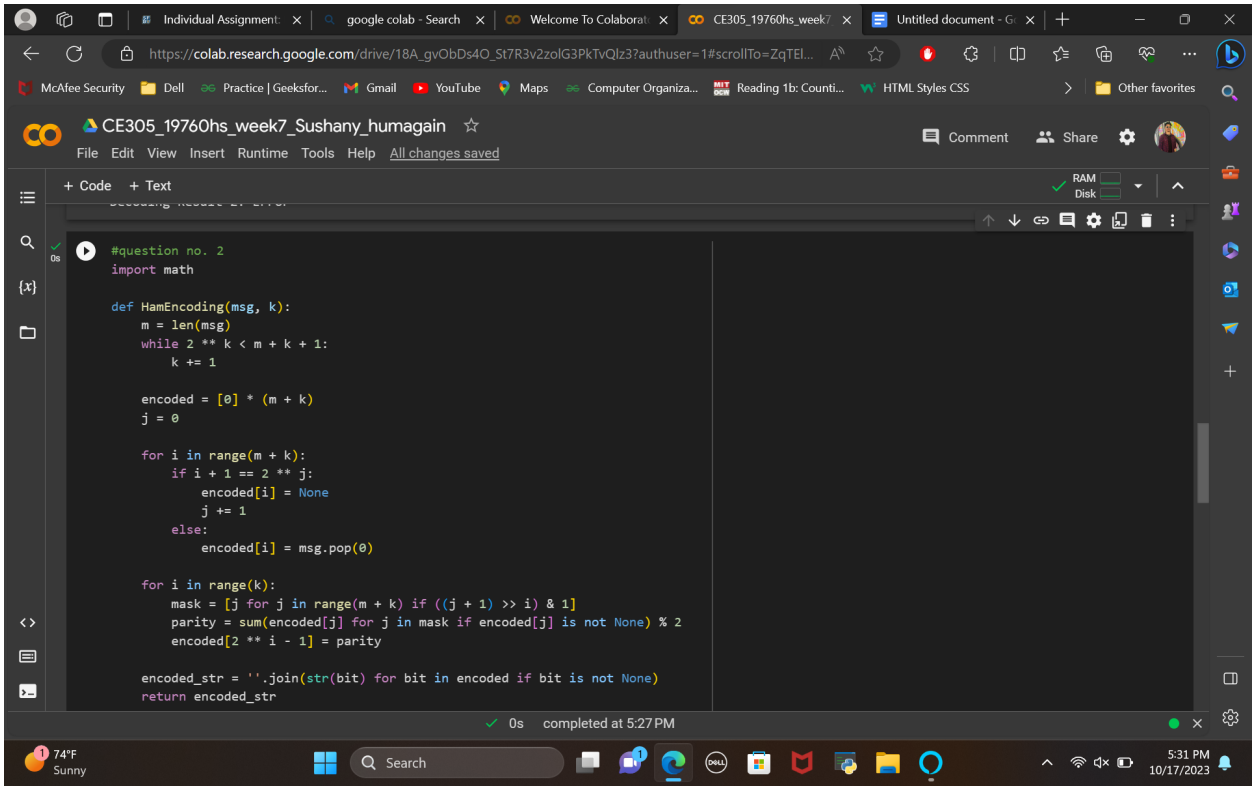
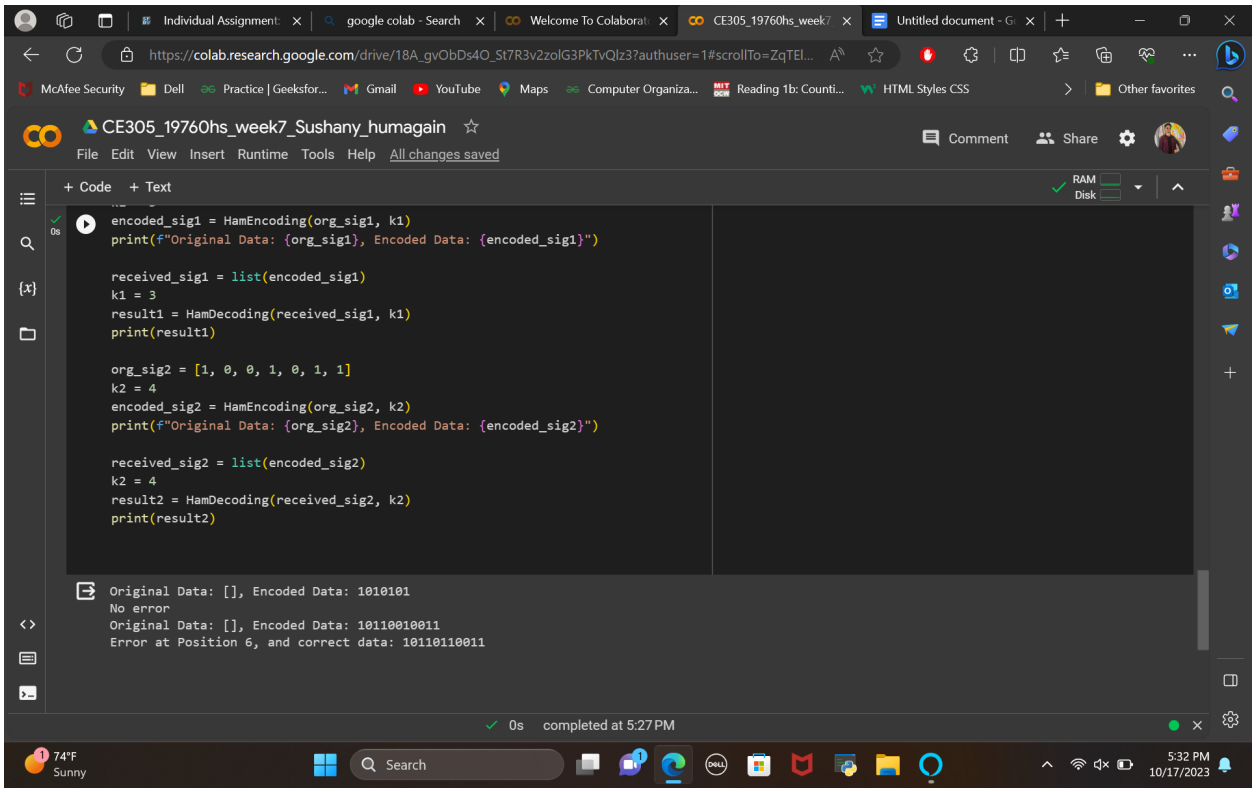

CE305_19760hs_week7_Sushany_humagain
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
#question no. 1
def xor(a, b):
result = []
for i in range(1, len(b)):
if a[i] == b[i]:
result.append('0')
else:
result.append('1')
return ''.join(result)
def mod2div(dividend, divisor):
pick = len(divisor)
tmp = dividend[0: pick]
while pick < len(dividend):
if tmp[0] == '1':
tmp = xor(divisor, tmp) + dividend[pick]
else:
tmp = xor('0'*pick, tmp) + dividend[pick]
pick += 1
if tmp[0] == '1':
tmp = xor(divisor, tmp)
0s completed at 5:27 PM
74°F
Sunny
Search
5:30 PM
10/17/2023


CE305_19760hs_week7_Sushany_humagain
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
return 'No error'
# Test cases
poly = '100101' # x^5 + x^2 + 1
org_sig1 = '1010' # original 4-bit binary data
encoded_output1 = encodeData(org_sig1, poly)
print("Encoded Output 1:", encoded_output1)
org_sig2 = '1100' # original 4-bit binary data
encoded_output2 = encodeData(org_sig2, poly)
print("Encoded Output 2:", encoded_output2)
received_sig1 = '1010 00111' # if receiving the data without error
decoding_result1 = decodeData(received_sig1, poly)
print("Decoding Result 1:", decoding_result1)
received_sig2 = '1010 01111' # if receiving the data with 1-bit error
decoding_result2 = decodeData(received_sig2, poly)
print("Decoding Result 2:", decoding_result2)
Encoded Output 1: 101000111
Encoded Output 2: 110011001
Decoding Result 1: No error
Decoding Result 2: Error
0s completed at 5:27 PM
74°F
Sunny
Search
5:31 PM
10/17/2023



```
#question no. 2
import math

def HamEncoding(msg, k):
    m = len(msg)
    while 2 ** k < m + k + 1:
        k += 1

    encoded = [0] * (m + k)
    j = 0

    for i in range(m + k):
        if i + 1 == 2 ** j:
            encoded[i] = None
            j += 1
        else:
            encoded[i] = msg.pop(0)

    for i in range(k):
        mask = [j for j in range(m + k) if ((j + 1) >> i) & 1]
        parity = sum(encoded[j] for j in mask if encoded[j] is not None) % 2
        encoded[2 ** i - 1] = parity

    encoded_str = ''.join(str(bit) for bit in encoded if bit is not None)
    return encoded_str
```



```
encoded_sig1 = HamEncoding(org_sig1, k1)
print(f"Original Data: {org_sig1}, Encoded Data: {encoded_sig1}")

received_sig1 = list(encoded_sig1)
k1 = 3
result1 = HamDecoding(received_sig1, k1)
print(result1)

org_sig2 = [1, 0, 0, 1, 0, 1, 1]
k2 = 4
encoded_sig2 = HamEncoding(org_sig2, k2)
print(f"Original Data: {org_sig2}, Encoded Data: {encoded_sig2}")

received_sig2 = list(encoded_sig2)
k2 = 4
result2 = HamDecoding(received_sig2, k2)
print(result2)
```

```
Original Data: [], Encoded Data: 1010101
No error
Original Data: [], Encoded Data: 10110010011
Error at Position 6, and correct data: 10110110011
```