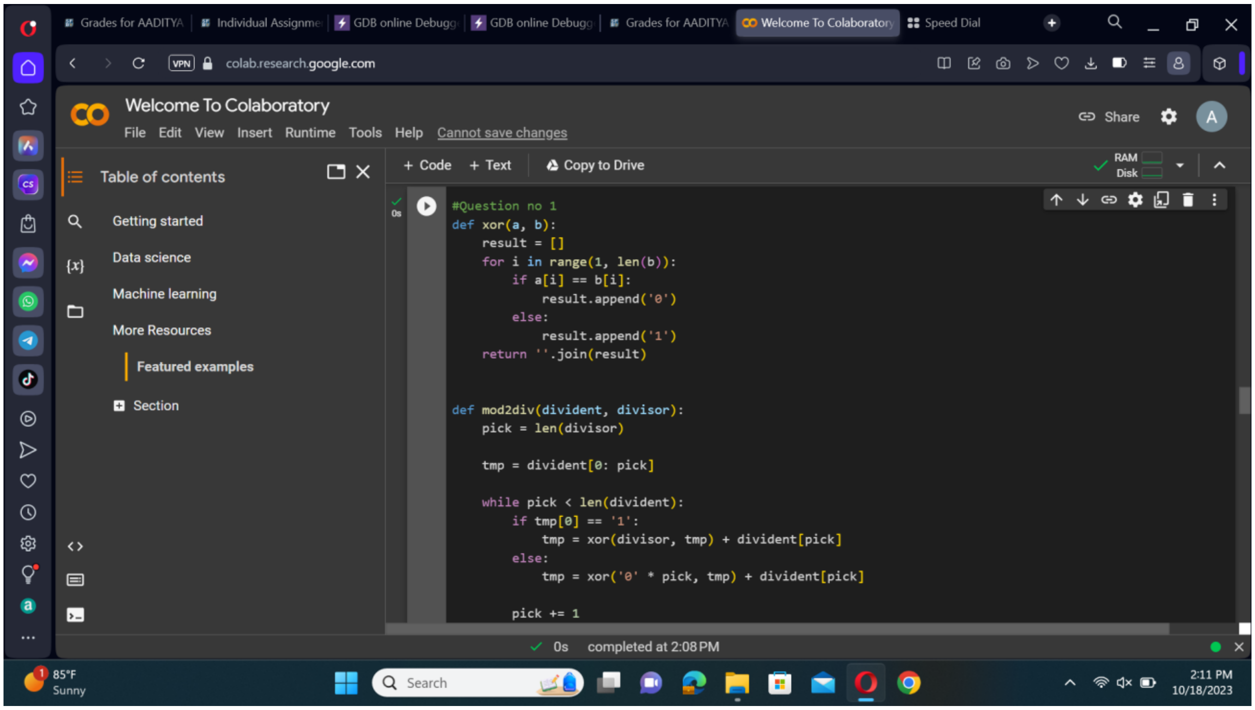
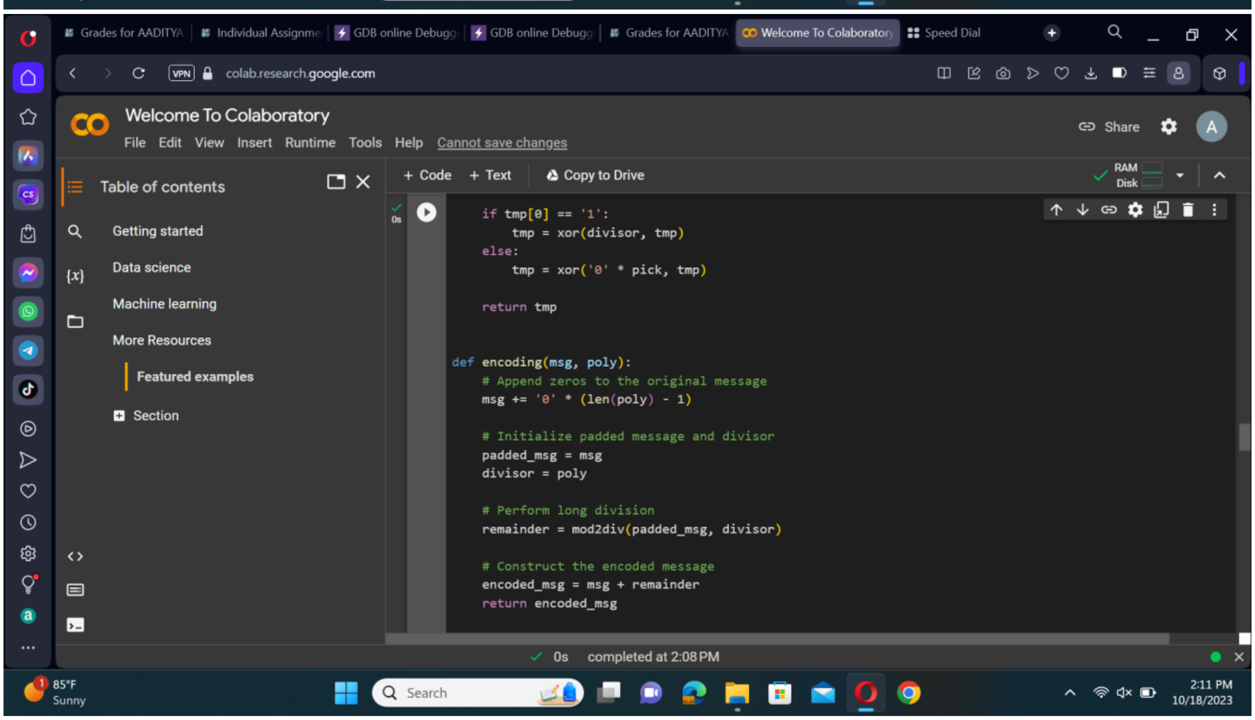
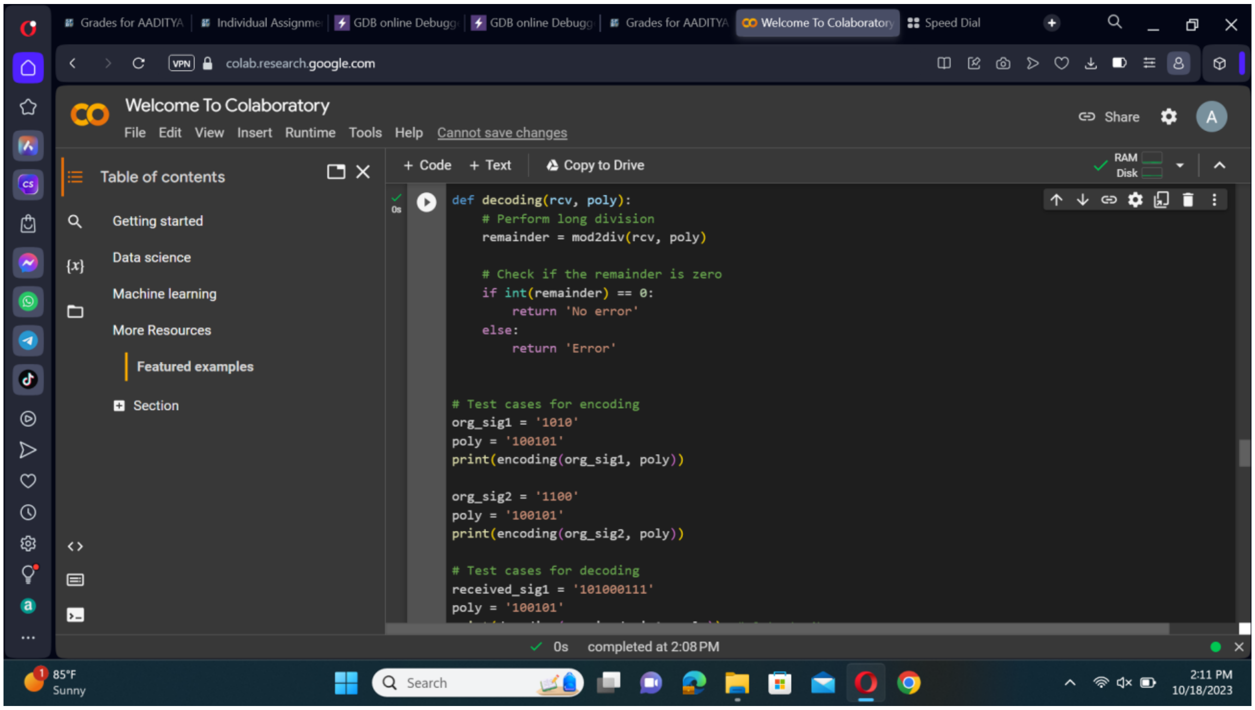
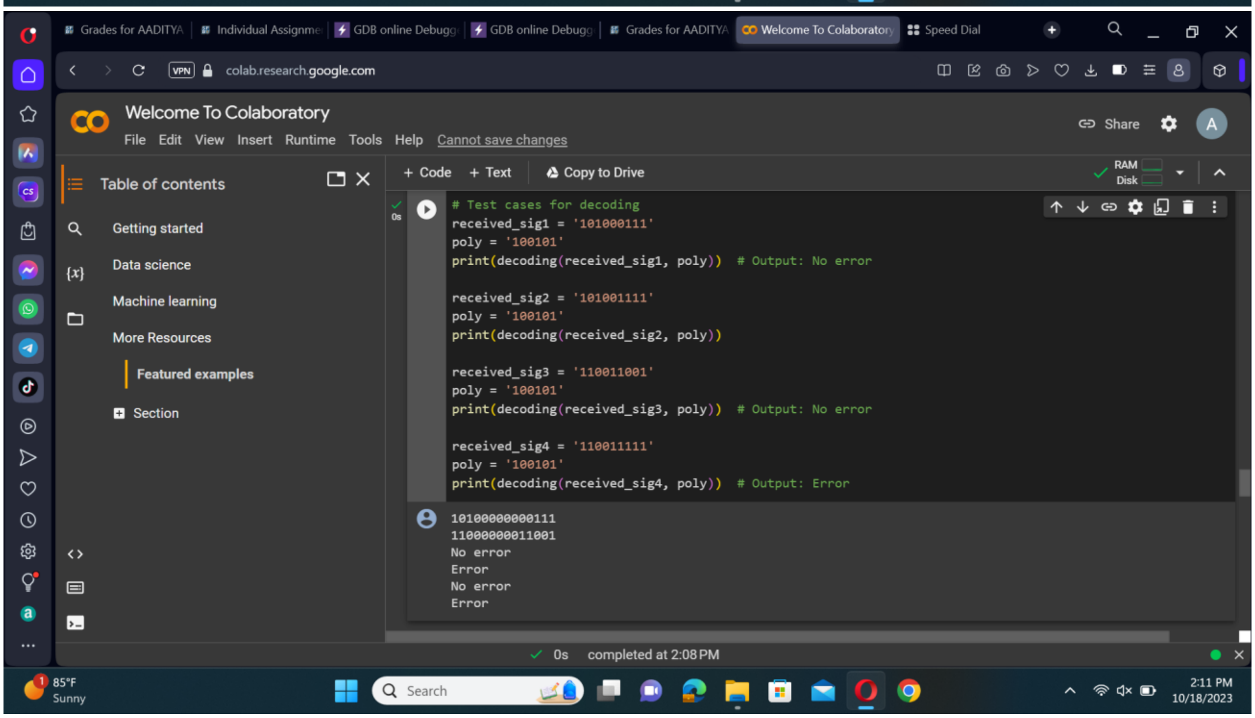
1. Cyclic Redundancy Check (CRC) is one of the popular coding and decoding techniques in the data transmitted over the network for error detection and correction. Given $x5 + x2 + 1$ as a CRC generation polynomial from International Telegraph and Telephone Consultative Committee (CCITT), write the encoding and decoding def functions in Python for the only 4-bits original binary data. The examples and testcases of the encoding and decoding processes are shown as follows for your programming. After that, discuss how many bits errors CRC can detect







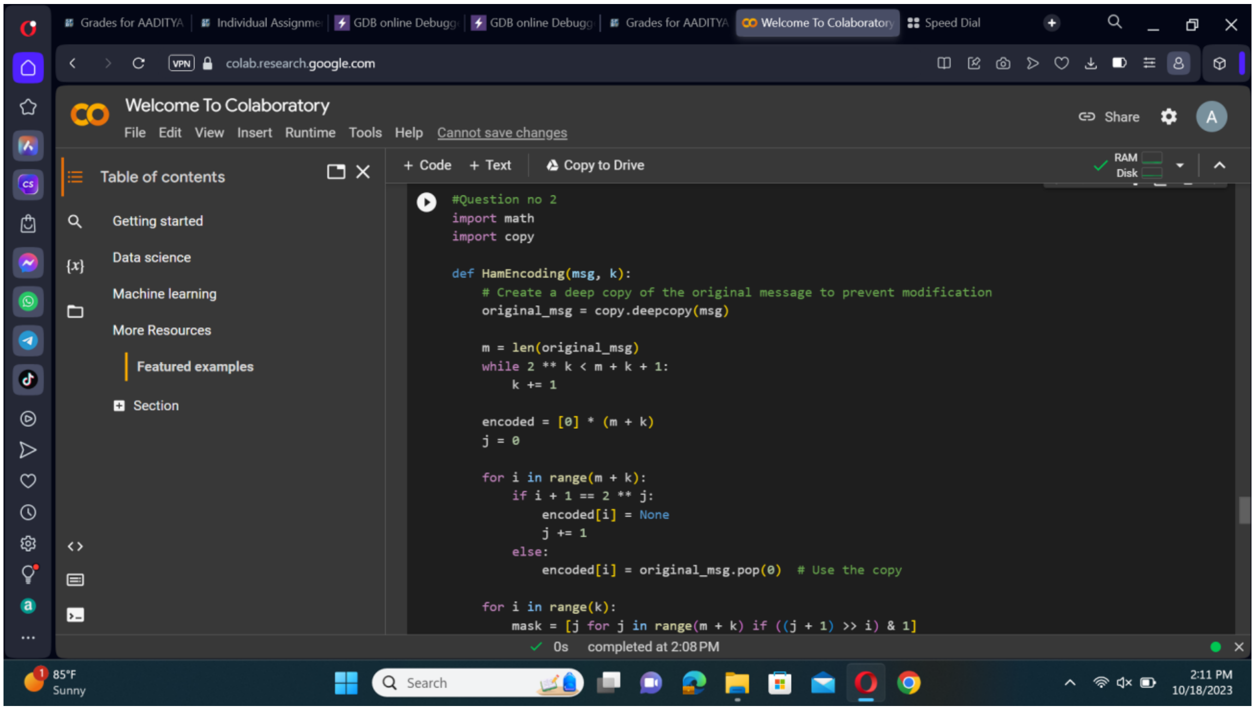
```python
def decoding(rcv, poly):
    # Perform long division
    remainder = mod2div(rcv, poly)

    # Check if the remainder is zero
    if int(remainder) == 0:
        return 'No error'
    else:
        return 'Error'


# Test cases for encoding
org_sig1 = '1010'
poly = '100101'
print(encoding(org_sig1, poly))

org_sig2 = '1100'
poly = '100101'
print(encoding(org_sig2, poly))

# Test cases for decoding
received_sig1 = '101000111'
poly = '100101'
```

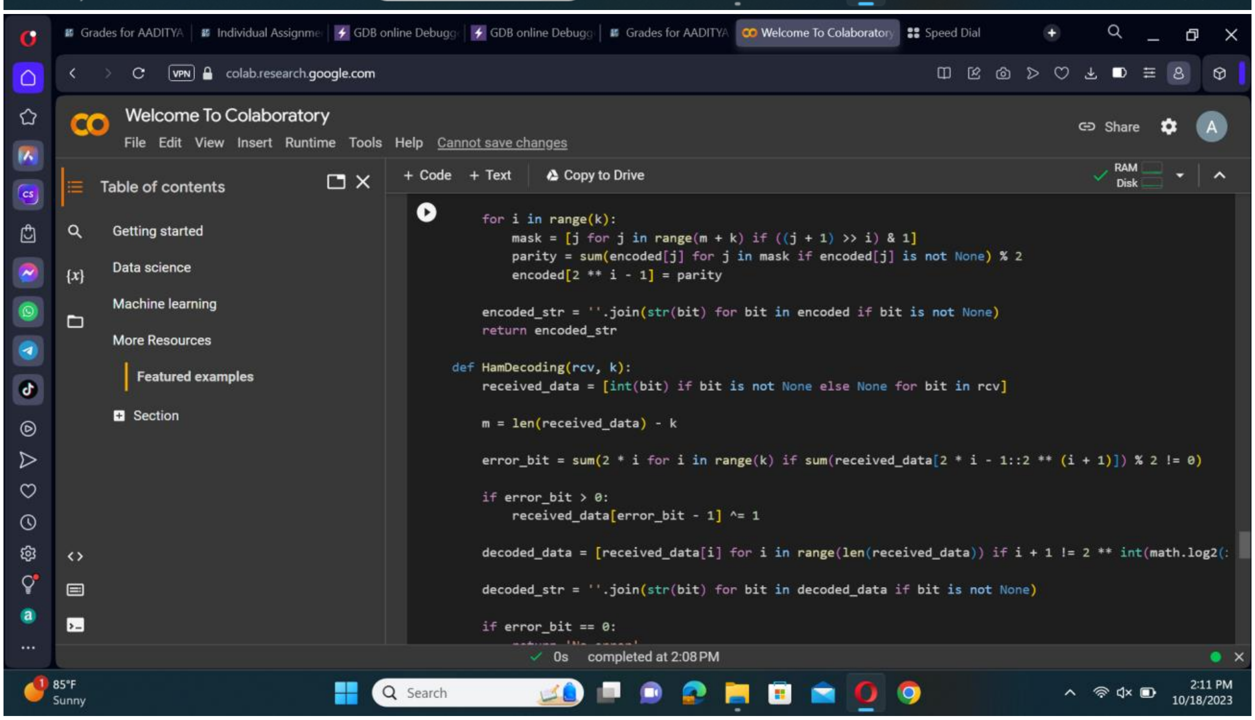


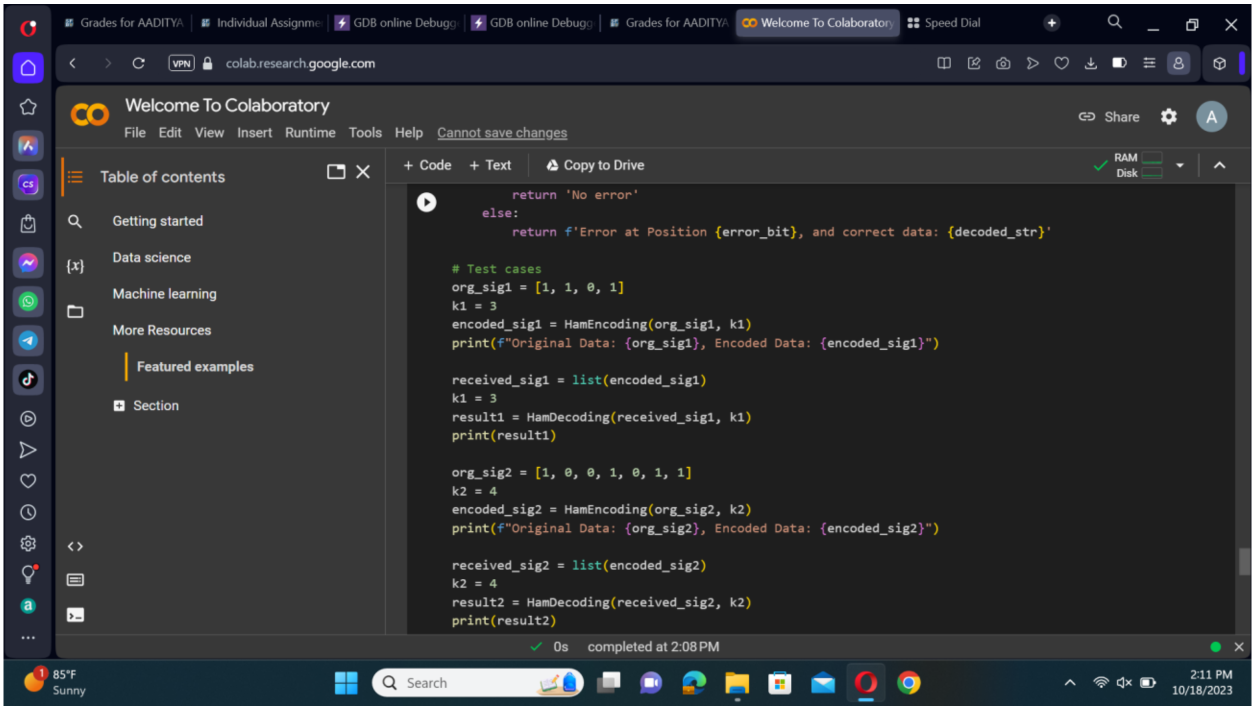
```python
# Test cases for decoding
received_sig1 = '101000111'
poly = '100101'
print(decoding(received_sig1, poly))  # Output: No error

received_sig2 = '101001111'
poly = '100101'
print(decoding(received_sig2, poly))

received_sig3 = '110011001'
poly = '100101'
print(decoding(received_sig3, poly))  # Output: No error

received_sig4 = '110011111'
poly = '100101'
print(decoding(received_sig4, poly))  # Output: Error
```

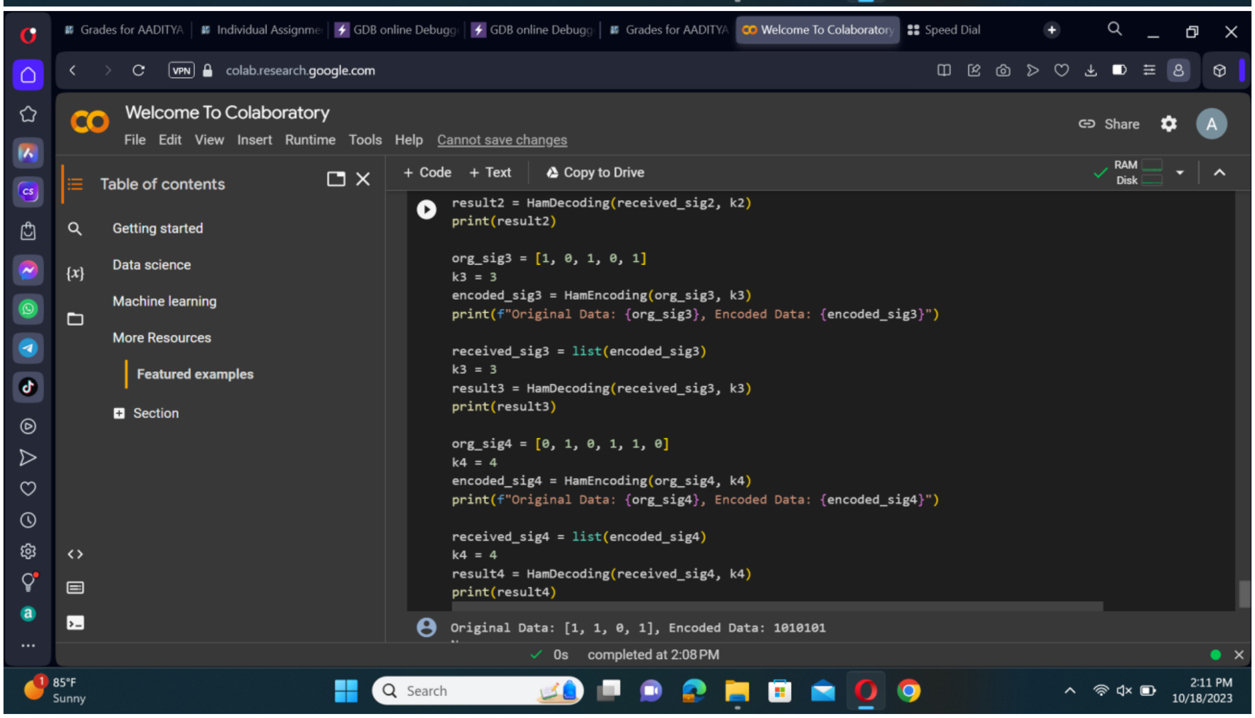
```
10100000000111
11000000011001
No error
Error
No error
Error
```

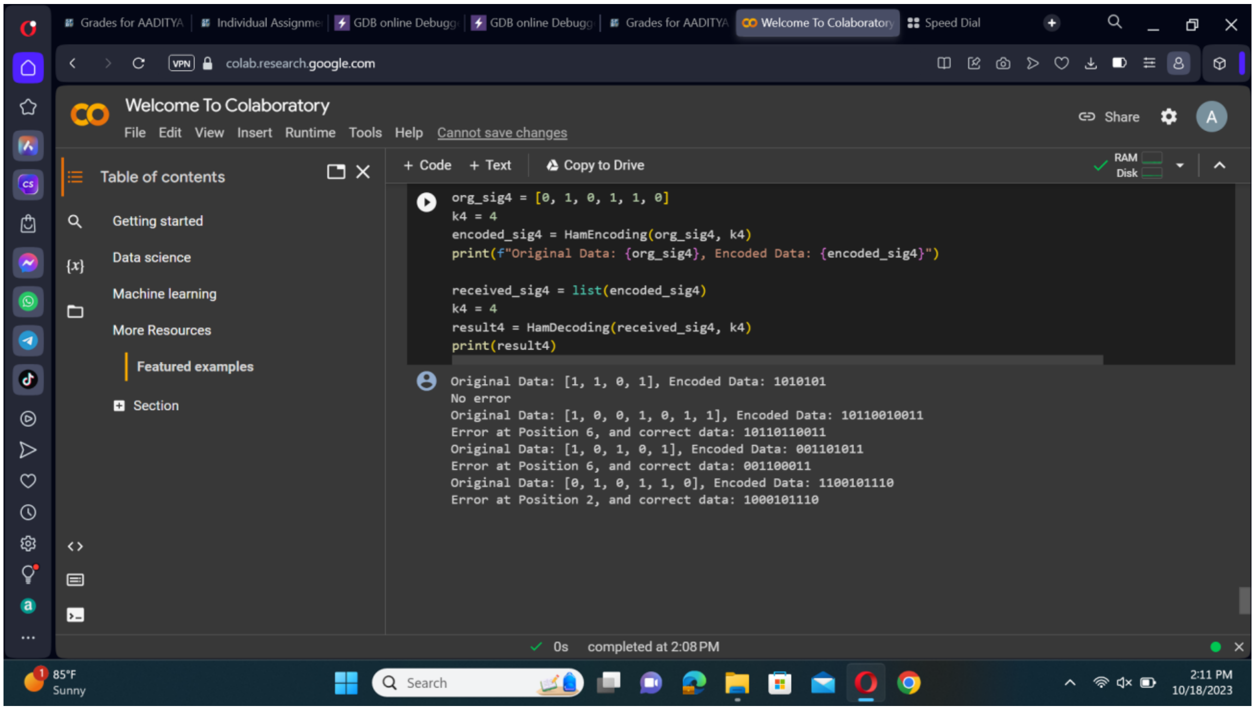
2. Hamming code is one important error correcting code in computer science and telecommunication as well. Standard Hamming code can only detect and correct a single bit error. Below is my solution for the problem






Welcome To Colaboratory
return 'No error'
else:
return f'Error at Position {error_bit}, and correct data: {decoded_str}'
# Test cases
org_sig1 = [1, 1, 0, 1]
k1 = 3
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)
0s completed at 2:08 PM


Welcome To Colaboratory
result2 = HamDecoding(received_sig2, k2)
print(result2)
org_sig3 = [1, 0, 1, 0, 1]
k3 = 3
encoded_sig3 = HamEncoding(org_sig3, k3)
print(f"Original Data: {org_sig3}, Encoded Data: {encoded_sig3}")
received_sig3 = list(encoded_sig3)
k3 = 3
result3 = HamDecoding(received_sig3, k3)
print(result3)
org_sig4 = [0, 1, 0, 1, 1, 0]
k4 = 4
encoded_sig4 = HamEncoding(org_sig4, k4)
print(f"Original Data: {org_sig4}, Encoded Data: {encoded_sig4}")
received_sig4 = list(encoded_sig4)
k4 = 4
result4 = HamDecoding(received_sig4, k4)
print(result4)
Original Data: [1, 1, 0, 1], Encoded Data: 1010101
0s completed at 2:08 PM


Welcome To Colaboratory
File Edit View Insert Runtime Tools Help Cannot save changes
Table of contents
Getting started
Data science
Machine learning
More Resources
Featured examples
Section
org_sig4 = [0, 1, 0, 1, 1, 0]
k4 = 4
encoded_sig4 = HamEncoding(org_sig4, k4)
print(f"Original Data: {org_sig4}, Encoded Data: {encoded_sig4}")
received_sig4 = list(encoded_sig4)
k4 = 4
result4 = HamDecoding(received_sig4, k4)
print(result4)
Original Data: [1, 1, 0, 1], Encoded Data: 1010101
No error
Original Data: [1, 0, 0, 1, 0, 1, 1], Encoded Data: 10110010011
Error at Position 6, and correct data: 10110110011
Original Data: [1, 0, 1, 0, 1], Encoded Data: 001101011
Error at Position 6, and correct data: 001100011
Original Data: [0, 1, 0, 1, 1, 0], Encoded Data: 1100101110
Error at Position 2, and correct data: 1000101110
0s completed at 2:08 PM
2:11 PM
10/18/2023