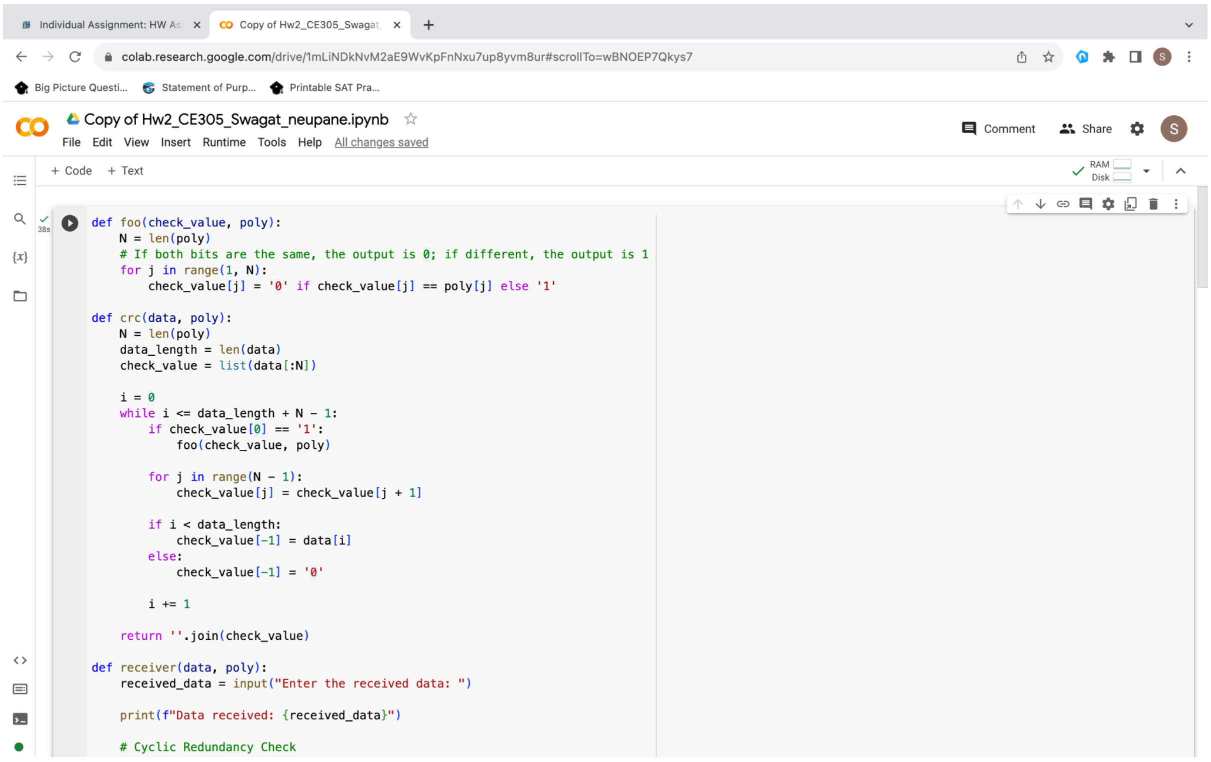
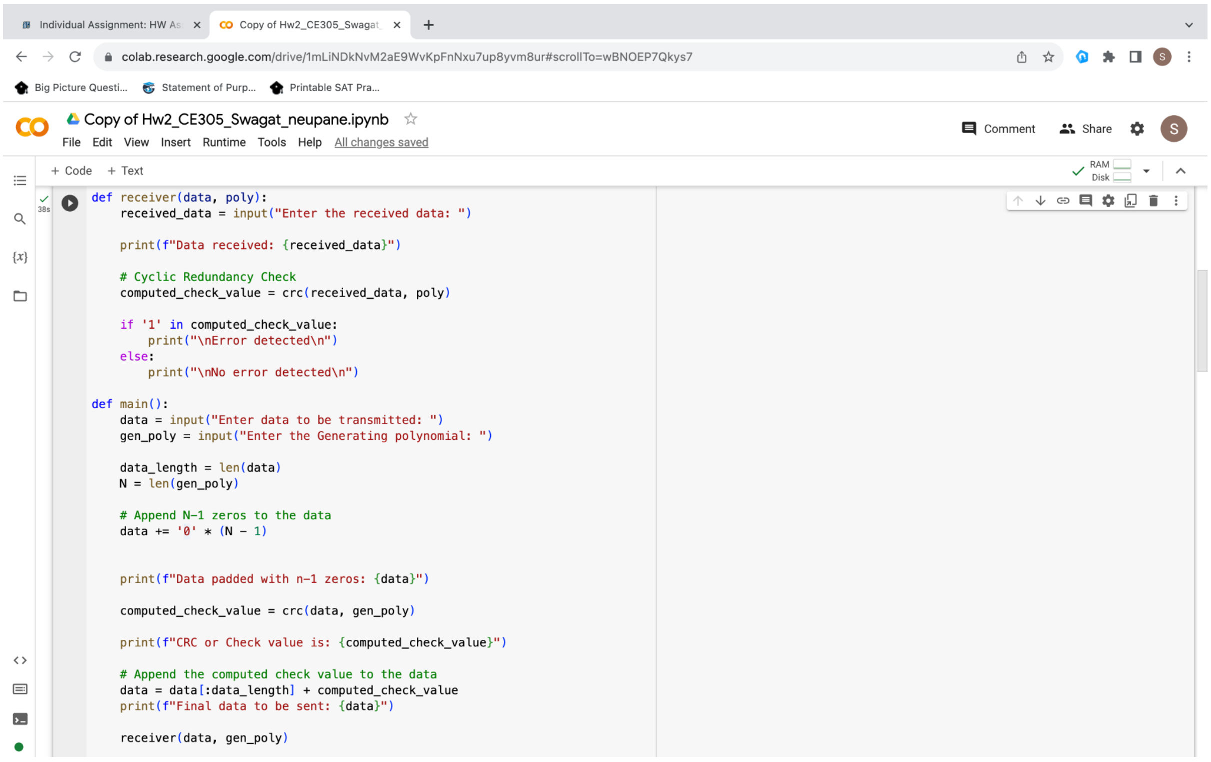
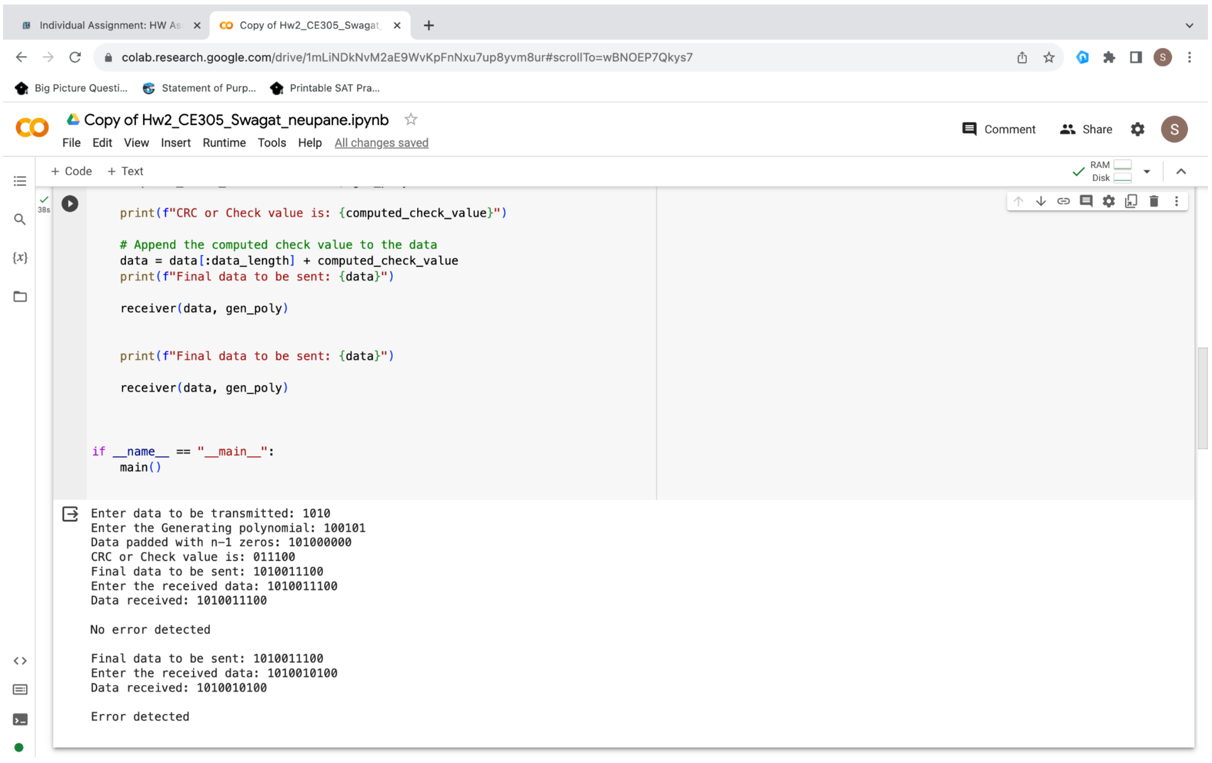
Swagat Neupane
CE305
Hw2


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=wBNOEP7Qkys7
Big Picture Questi...
Statement of Purp...
Printable SAT Pra...
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
38s

```python
def foo(check_value, poly):
    N = len(poly)
    # If both bits are the same, the output is 0; if different, the output is 1
    for j in range(1, N):
        check_value[j] = '0' if check_value[j] == poly[j] else '1'

def crc(data, poly):
    N = len(poly)
    data_length = len(data)
    check_value = list(data[:N])

    i = 0
    while i <= data_length + N - 1:
        if check_value[0] == '1':
            foo(check_value, poly)

        for j in range(N - 1):
            check_value[j] = check_value[j + 1]

        if i < data_length:
            check_value[-1] = data[i]
        else:
            check_value[-1] = '0'

        i += 1

    return ''.join(check_value)

def receiver(data, poly):
    received_data = input("Enter the received data: ")

    print(f"Data received: {received_data}")

    # Cyclic Redundancy Check
```


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=wBNOEP7Qkys7
Big Picture Questi...
Statement of Purp...
Printable SAT Pra...
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
38s

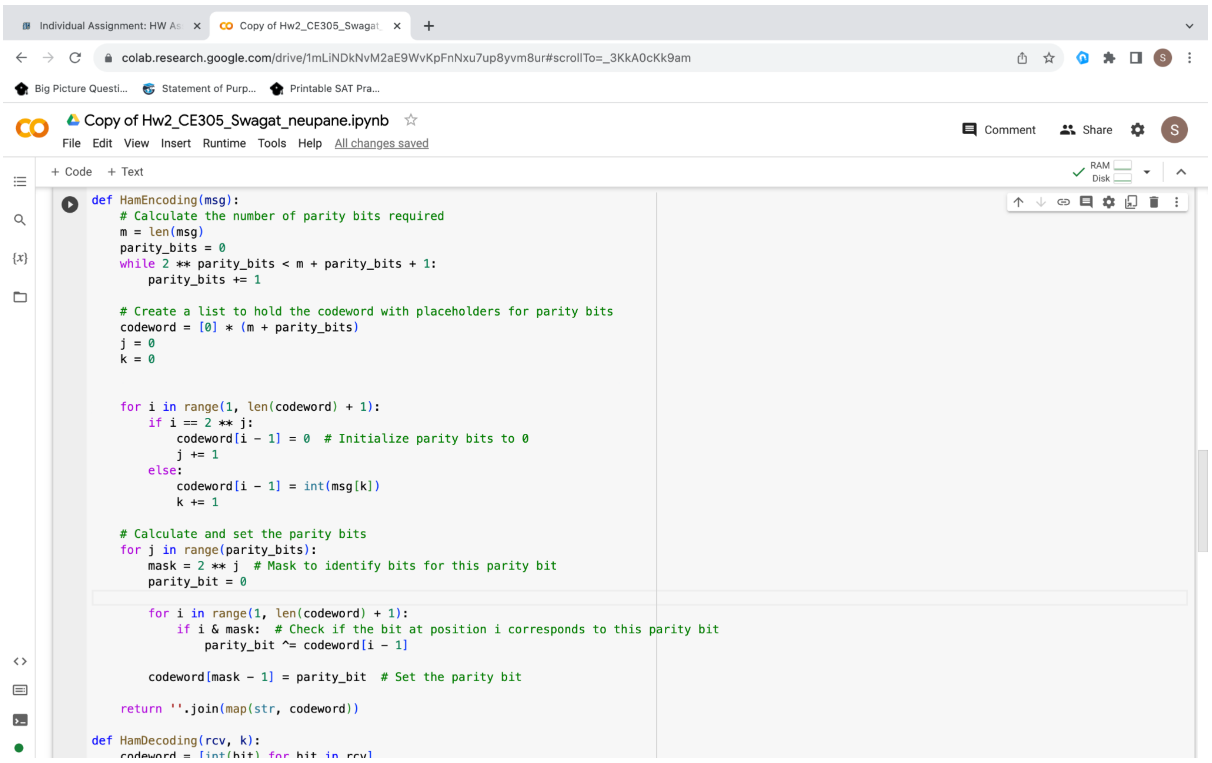
```python
def receiver(data, poly):
    received_data = input("Enter the received data: ")

    print(f"Data received: {received_data}")

    # Cyclic Redundancy Check
    computed_check_value = crc(received_data, poly)

    if '1' in computed_check_value:
        print("\nError detected\n")
    else:
        print("\nNo error detected\n")

def main():
    data = input("Enter data to be transmitted: ")
    gen_poly = input("Enter the Generating polynomial: ")

    data_length = len(data)
    N = len(gen_poly)

    # Append N-1 zeros to the data
    data += '0' * (N - 1)


    print(f"Data padded with n-1 zeros: {data}")

    computed_check_value = crc(data, gen_poly)

    print(f"CRC or Check value is: {computed_check_value}")

    # Append the computed check value to the data
    data = data[:data_length] + computed_check_value
    print(f"Final data to be sent: {data}")

    receiver(data, gen_poly)
```


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=wBNOEP7Qkys7
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text

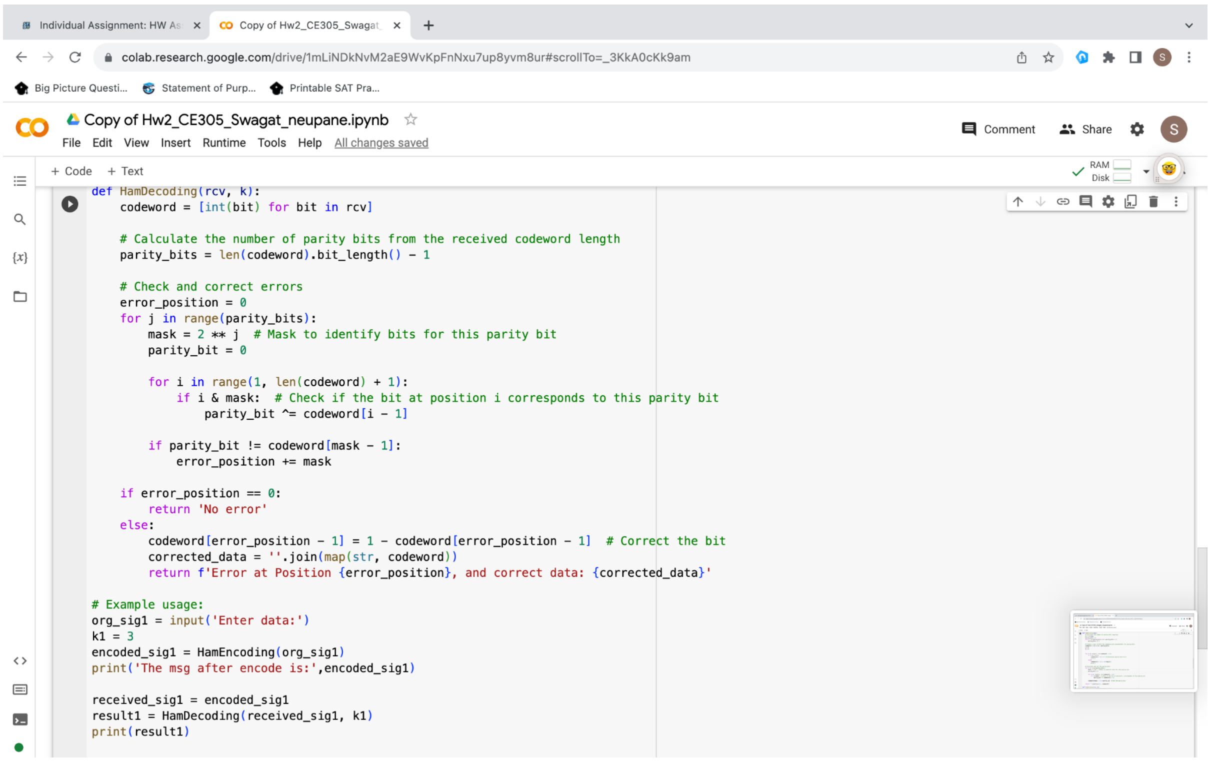
```
    print(f"CRC or Check value is: {computed_check_value}")

    # Append the computed check value to the data
    data = data[:data_length] + computed_check_value
    print(f"Final data to be sent: {data}")

    receiver(data, gen_poly)


    print(f"Final data to be sent: {data}")

    receiver(data, gen_poly)



if __name__ == "__main__":
    main()
```

```
Enter data to be transmitted: 1010
Enter the Generating polynomial: 100101
Data padded with n-1 zeros: 101000000
CRC or Check value is: 011100
Final data to be sent: 1010011100
Enter the received data: 1010011100
Data received: 1010011100

No error detected

Final data to be sent: 1010011100
Enter the received data: 1010010100
Data received: 1010010100

Error detected
```

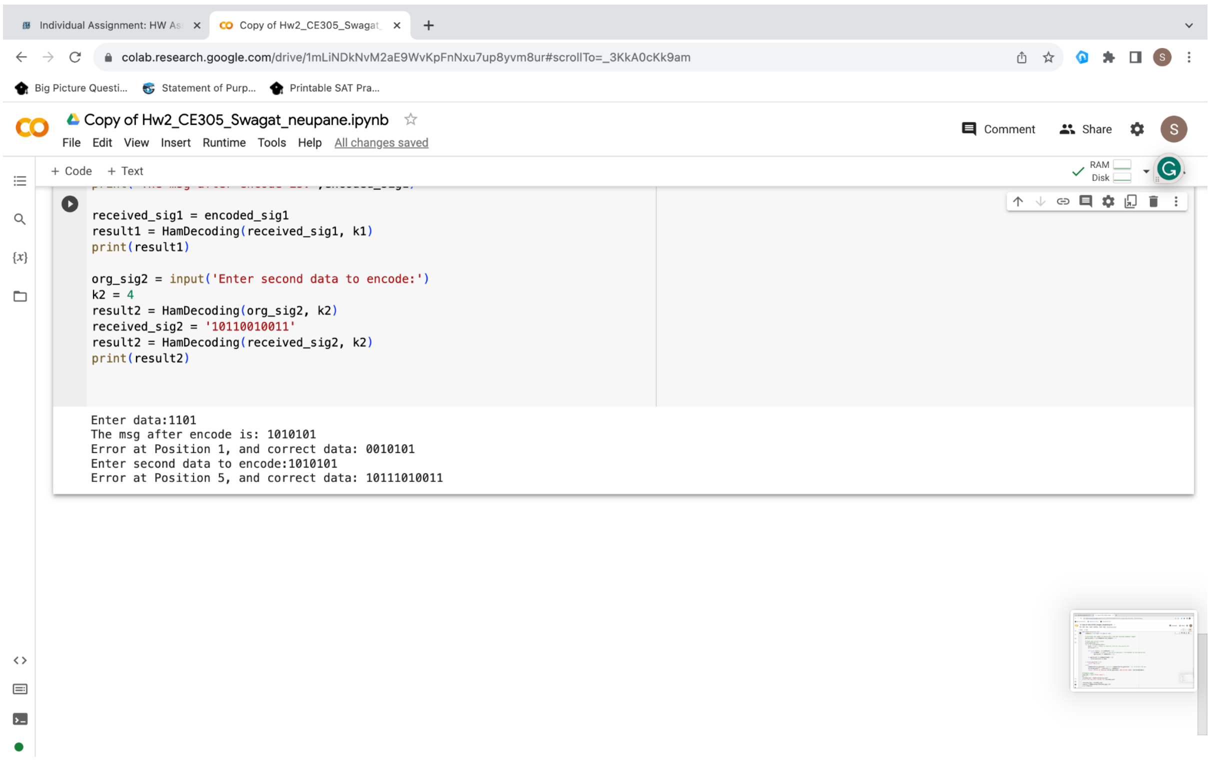
Solution-2:


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=_3KkA0cKk9am
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text

```
def HamEncoding(msg):
    # Calculate the number of parity bits required
    m = len(msg)
    parity_bits = 0
    while 2 ** parity_bits < m + parity_bits + 1:
        parity_bits += 1

    # Create a list to hold the codeword with placeholders for parity bits
    codeword = [0] * (m + parity_bits)
    j = 0
    k = 0


    for i in range(1, len(codeword) + 1):
        if i == 2 ** j:
            codeword[i - 1] = 0  # Initialize parity bits to 0
            j += 1
        else:
            codeword[i - 1] = int(msg[k])
            k += 1

    # Calculate and set the parity bits
    for j in range(parity_bits):
        mask = 2 ** j  # Mask to identify bits for this parity bit
        parity_bit = 0

        for i in range(1, len(codeword) + 1):
            if i & mask:  # Check if the bit at position i corresponds to this parity bit
                parity_bit ^= codeword[i - 1]

        codeword[mask - 1] = parity_bit  # Set the parity bit

    return ''.join(map(str, codeword))

def HamDecoding(rcv, k):
```


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=_3KkA0cKk9am
Big Picture Questi...
Statement of Purp...
Printable SAT Pra...
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

```python
def HamDecoding(rcv, k):
    codeword = [int(bit) for bit in rcv]

    # Calculate the number of parity bits from the received codeword length
    parity_bits = len(codeword).bit_length() - 1

    # Check and correct errors
    error_position = 0
    for j in range(parity_bits):
        mask = 2 ** j  # Mask to identify bits for this parity bit
        parity_bit = 0

        for i in range(1, len(codeword) + 1):
            if i & mask:  # Check if the bit at position i corresponds to this parity bit
                parity_bit ^= codeword[i - 1]

        if parity_bit != codeword[mask - 1]:
            error_position += mask

    if error_position == 0:
        return 'No error'
    else:
        codeword[error_position - 1] = 1 - codeword[error_position - 1]  # Correct the bit
        corrected_data = ''.join(map(str, codeword))
        return f'Error at Position {error_position}, and correct data: {corrected_data}'

# Example usage:
org_sig1 = input('Enter data:')
k1 = 3
encoded_sig1 = HamEncoding(org_sig1)
print('The msg after encode is:',encoded_sig1)

received_sig1 = encoded_sig1
result1 = HamDecoding(received_sig1, k1)
print(result1)
```


Individual Assignment: HW As
Copy of Hw2_CE305_Swagat
colab.research.google.com/drive/1mLiNDkNvM2aE9WvKpFnNxu7up8yvm8ur#scrollTo=_3KkA0cKk9am
Big Picture Questi...
Statement of Purp...
Printable SAT Pra...
Copy of Hw2_CE305_Swagat_neupane.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

```python
received_sig1 = encoded_sig1
result1 = HamDecoding(received_sig1, k1)
print(result1)

org_sig2 = input('Enter second data to encode:')
k2 = 4
result2 = HamDecoding(org_sig2, k2)
received_sig2 = '10110010011'
result2 = HamDecoding(received_sig2, k2)
print(result2)
```

```
Enter data:1101
The msg after encode is: 1010101
Error at Position 1, and correct data: 0010101
Enter second data to encode:1010101
Error at Position 5, and correct data: 10111010011
```