Question 1

Assign2.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

```python
#Answer of 1
def crc_encode(data, poly):
    n = len(bin(poly)) - 3
    data = int(data, 2) << n
    divisor = poly << (n - 1)
    mask = 1 << (n + 4)
    dividend = data << 4

    for _ in range(n + 4):
        if dividend & mask:
            dividend ^= divisor
        divisor >>= 1
        mask >>= 1

    return format((data | dividend) >> 4, f'0{len(bin(data)) + n}b')[4:]

def crc_decode(received, poly):
    n = len(bin(poly)) - 3
    received = int(received, 2)
    divisor = poly << (n - 1)
    mask = 1 << (n + 4)
    dividend = received << 4

    for _ in range(n + 4):
        if dividend & mask:
            dividend ^= divisor
        divisor >>= 1
        mask >>= 1

    return bin(dividend)[-n-4:].zfill(n+4) == '0' * (n + 4)


# Example usage
d = '1101'  # 4-bit binary data
p = 0x04C11DB7  # CRC-32 polynomial

e = crc_encode(d, p)
print(f'Encoded Data: {e}')

r = e  # Simulating received data
```

Assign2.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

```python
error_detected = not crc_decode(r, p)

if error_detected:
    print('Error detected!')
else:
    print('No error detected.')
```

```
Encoded Data: 000000111101000111000111011011110101000110111111010100
Error detected!
```

Question 2

Assign2.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

```python
#Answer of 2
def hamming_encode(data):
    m = len(data)
    k = 1
    while 2**k < m + k + 1:
        k += 1

    parity_positions = [2**i for i in range(k)]

    encoded_data = [0] * (m + k)
    j = 0
    for i in range(1, m + k + 1):
        if i not in parity_positions:
            encoded_data[i - 1] = int(data[j])
            j += 1

    for i in range(k):
        mask = 2**i
        ones_count = sum(encoded_data[j - 1] for j in range(1, m + k + 1) if j & mask)
        encoded_data[parity_positions[i] - 1] = ones_count % 2

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

original_data = '1001'
encoded_data = hamming_encode(original_data)
print(f'Encoded Data: {encoded_data}')


# checking the bit position for extra parity
def calculate_parity_positions(k):
    return [2**i for i in range(k)]

num_parity_bits = 4
parity_positions = calculate_parity_positions(num_parity_bits)

print(f"The positions of the extra parity bits for k={num_parity_bits} are: {parity_positions}")


# creating a table
def p(encoded_data, pos):
    mask = 2**pos
```

Assign2.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

```python
    covered_bits = [int(encoded_data[i - 1]) for i in range(1, len(encoded_data) + 1) if i & mask]
    return sum(covered_bits) % 2

def h(data):
    m = len(data)
    k = 1
    while 2**k < m + k + 1:
        k += 1

    p_positions = [2**i for i in range(k)]

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

    for i in range(1, m + k + 1):
        if i not in p_positions:
            encoded_data[i - 1] = int(data[j])
            j += 1

    for i in range(k):
        encoded_data[p_positions[i] - 1] = p(encoded_data, i)

    return encoded_data

def t(data, encoded_data):
    m = len(data)
    k = len(encoded_data) - m

    print("Original Data   |  Parity Bits  |  Encoded Data")
    print("-" * 45)

    j = 0
    for i in range(1, m + k + 1):
        if i not in [2**x for x in range(k)]:
            print(f"       {data[j]}        |               |       {encoded_data[i-1]}")
            j += 1
        else:
            print("                  |       {}       |".format(encoded_data[i-1]))

o = '1001'
e = h(o)
```


```python
e = h(o)
```


```python
t(o, e)
```

```
Encoded Data: 0011001
The positions of the extra parity bits for k=4 are: [1, 2, 4, 8]
Original Data   |  Parity Bits  |  Encoded Data
---------------------------------------------
                  |       0       |
                  |       0       |
       1         |               |       1
                  |       1       |
       0         |               |       0
       0         |               |       0
       1         |               |       1
```