07/8/2025 Practical VI

Error Correction at Data Link Layer:

Hamming code is a set of error-detection codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Sender Program

1. Input to sender file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on the binary data and add redundant bits to it.
3. Save this output in a file called channel.

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

def encode_hamming(data):
    m= len(data)
    r=0
    while(2**r<m+r+1):
    r+=1
    j,k,res = 0,1,''
    for i in range (1, m+r+1):
        if i== 2**j:
            res+='0'
            j+=1
        else:
            res+= data[-k]
            k+=1
    arr= list(res[::-1])
    n= len(arr)
    for i in range(r):
        val=0
        for j in range(1,n+1):
            if j & (2**i) = (2**i):
                val ^= int(arr[-j])
        arr[-(2**i)] = str(val)
    return ''.join(arr)

def sender(input, output_file="channel"):
    binary_data = text_binary(input)
    encoded_data = encode_hamming(binary_data)
    with open(output_file,"w") as f:
        f.write(encoded_data)

    print("Input text:  ", ~~input_text~~)
    print("Binary data:  ", binary_data)
    print("Hamming Encoded: ", encoded_data)
    print(f"Encoded data saved in '{output_file}'.")
```

Input:

```
Enter text to send: Hi
```

Output:

```
Input text: Hi
Binary Data: 0100100001101001
Hamming Encoded: 010010000011001000100
Encoded data saved in 'channel'.
```

Receiver Program:

1. Receiver program should read the input from channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else, remove the redundant bits and convert the binary data to ASCII and display the output.

```
def detect_error (arr, r):
    n = len (arr)
    res = 0
    for i in range(r):
        val = 0
        for j in range (1, n+1):
            if j & (2**i) == (2**i):
                val ^= int (arr[-j])
        if val == 1:
            res += 2**i
    return res

def remove_bits (arr, r):
    n = len(arr)
    data_bits = []
    j = 0
    for i in range (1, n+1):
        if i != 2**j:
            data_bits.append (arr[-i])
        else:
            j += 1
    return "".join (data_bits [::-1])

def binary_text (binary_data):
    char = []
    for i in range (0, len (binary_data), 8):
        byte = binary_data [i: i+8]
    if len(byte)==8:
        chars.append(chr(int(byte,2)))
 return ''.join(chars)

def receiver(input_file="channel"):
  with open(input_file,"r") as f:
     encoded_data=f.read().strip()
  n=len(encoded_data)
  r=0
  while (2**r<n+1):
     r+=1
  error_pos=detect_error(encoded_data,r)
  if error_pos==0:
     print("No errors detected in received data.")
     data_bits=remove_bits(encoded_data,r)
     text=binary_text(data_bits)
     print("Recovered Binary:",data_bits)
     print("Recovered Text:",text)
  else:
     print(f"Error detected at bit position: {error_pos}")
```

Input:

Binary encoded data from channel.txt

Output:

```
No errors detected in received data.
Recovered binary: 0100100001101001
Recovered Text: Hi
```

[illegible] 4/9/18

Result:

Therefore, the code was encoded and decoded successfully.