Exp: 06 Hamming code

Aim: Write a program to implement error detection and correction using Hamming code Concept.

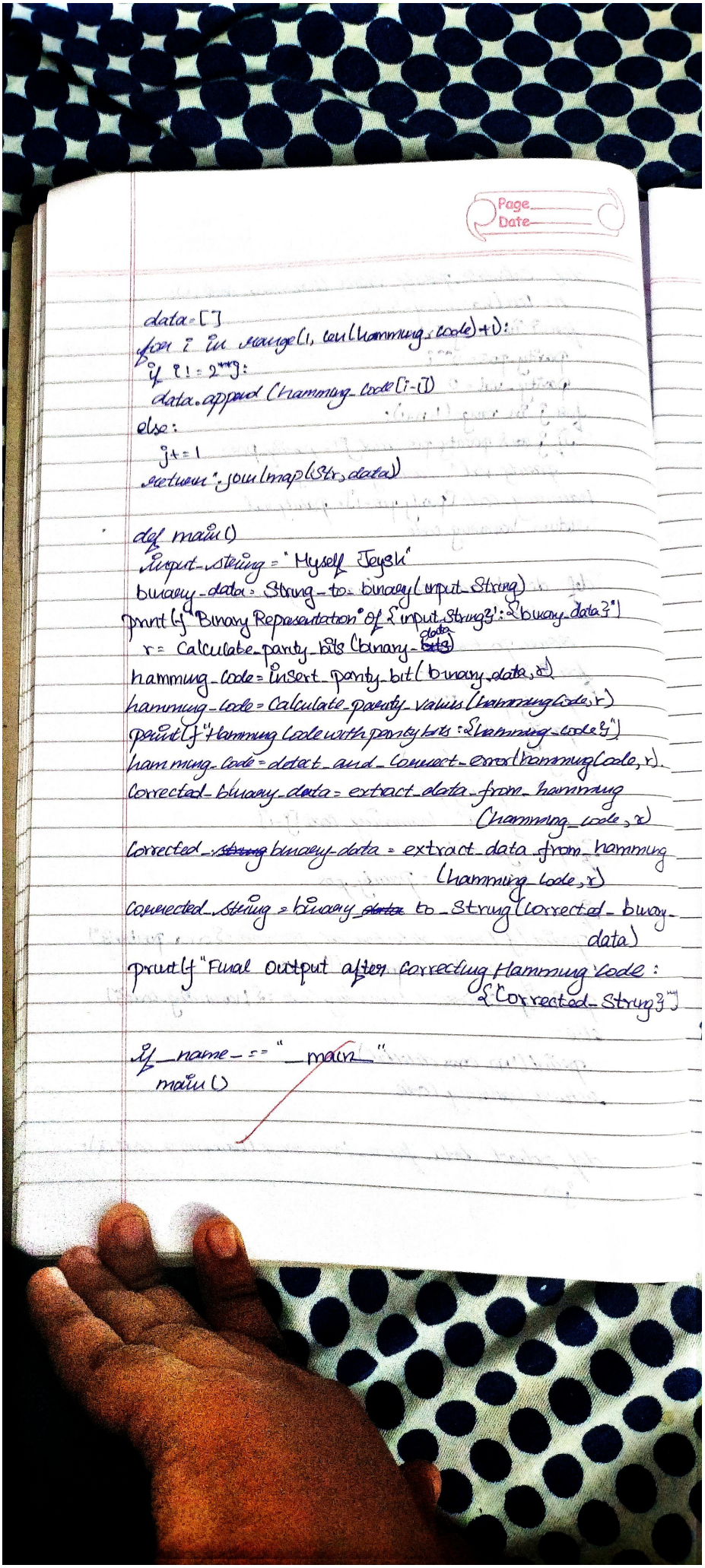
Program:

```
def string_to_binary(input_string):
return ''.join(format(ord(c), '08b') for c in input_string)
def binary_to_string(binary_data):
  chars = []
for i in range(0, len(binary_data), 8):
  byte = binary_data[i:i+8]
  chars.append(chr(int(byte, 2)))
return ''.join(chars)
def calculate_parity_bits(data):
  n = len(data)
  r = 0
while(2**r)<(n+r+1):
  r+=1
  return r
def insert_parity_bits(data, r):
  n = len(data)
  j = 0
  k = 0
  m = n+r
  hamming_code = []
if i==2**j:
  hamming_code.append(0)
  j+=1
else:
  hamming_code.append(int(data[k]))
  k+=1
return hamming_code
```

```
def calculate_parity_values(hamming_code, r):
    n = len(hamming_code)
    for i in range(r):
        parity_pos = 2**i
        parity_val = 0
        for j in range(1, n+1):
            if j & parity_pos and j != parity_pos:
                parity_val ^= hamming_code[j-1]
        hamming_code[parity_pos-1] = parity_val
    return hamming_code

def detect_and_correct(hamming_code, r):
    n = len(hamming_code)
    error_position = 0
    for i in range(r):
        parity_pos = 2**i
        parity_val = 0
        for j in range(1, n+1):
            if j & parity_pos:
                parity_val ^= hamming_code[j-1]
        if parity_val != 0:
            error_position += parity_pos
    if error_position:
        print(f"Error detected at position: {error_position}")
        hamming_code[error_position-1] ^= 1
        print(f"Corrected Hamming Code: {hamming_code}")
    else:
        print("No error detected")
    return hamming_code

def extract_data_from_hamming(hamming_code, r):
    j = 0
```



```
data = []
for i in range(1, len(hamming_code)+1):
    if i != 2**j:
        data.append(hamming_code[i-1])
else:
    j += 1
return ''.join(map(str, data))

def main()
    input_string = "Myself Jeyshi"
    binary_data = String_to_binary(input_String)
print(f"Binary Representation of {input_string}: {binary_data}")
    r = Calculate_parity_bits(binary_data)
    hamming_code = Insert_parity_bit(binary_data, r)
    hamming_code = Calculate_parity_values(hammingCode, r)
    print(f"Hamming Code with parity bits: {hamming_code}")
    hamming_code = detect_and_Correct_error(hammingCode, r)
    Corrected_binary_data = extract_data_from_hamming
                                (hamming_code, r)
    Corrected_string binary_data = extract_data_from_hamming
                                (hamming_code, r)
    Corrected_string = binary_data to_String(Corrected_binary_
                                data)
    print(f"Final output after correcting Hamming code:
                                {Corrected_String}")

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

Output :

Binary Representation of 'Myself Sayshi' :

0100110101111001011100110101101110001100111000100000001001
0101001100101011110010111001011010000.

Enter the bit position : 17.

Error detected at position : 17.

Corrected hamming code :-

[0,1,0,1,1,0,0,0,1,1,0,1,0,1,1,0,1,1,0,0,1,0, 1,1,1,0,0,1,1,0,1,1,
0,1,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1
0,0,1,0,1,0,1,1,0,1,1,0,0,0,1,1,0,0,1,1,0,0,0,1,0,0,0,0,1,0,0,1,0,1,1,0,1,1,0,1,0,0,0,0].
0,0,1,1,0,0,1,0,1,0,1,1,1,1,0,0,1,0,1,1,1,0,0,1,1,0,1,1,0,

Code execution Successful!

Result: The program is executed succesfully.