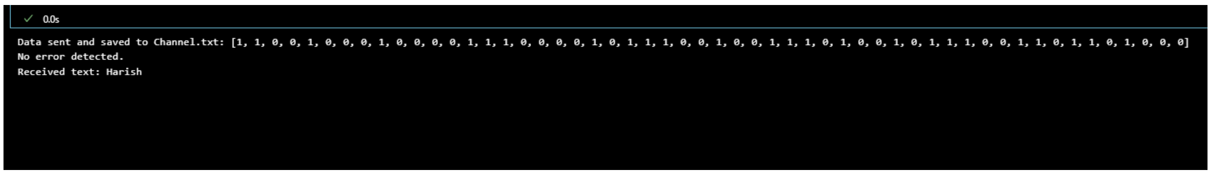
```python
def text_to_binary(text):
    binary_data = ''
    for char in text:
        binary_data += format(ord(char), '08b')  # Convert to 8-bit binary
    return binary_data

def calculate_redundant_bits(m):
    for i in range(m):
        if (2**i >= m + i + 1):
            return i

def generate_hamming_code(data):
    m = len(data)
    r = calculate_redundant_bits(m)
    arr = [0] * (m + r)
    j = 0
    for i in range(1, m + r + 1):
        if i == 2**j:  # Reserve positions for redundant bits
            j += 1
        else:
            arr[i - 1] = int(data[i - j - 1])  # Fill data bits
    for i in range(r):
        arr[(2**i) - 1] = calculate_parity(arr, i, m, r)  # Set parity bits
    return arr

def calculate_parity(arr, i, m, r):
    val = 0
    for j in range(1, m + r + 1):
        if j & (2**i) == (2**i):  # Check parity bit positions
            val ^= arr[j - 1]
    return val

def sender():
    # Input data
    text = "Giri"
    binary_data = text_to_binary(text)

    hamming_data = generate_hamming_code(binary_data)

    with open("Channel.txt", "w") as f:
        f.write(''.join(map(str, hamming_data)))

    print("Data sent and saved to Channel.txt:", hamming_data)

def read_channel():
    with open("Channel.txt", "r") as f:
        return f.read()
```

```python
def detect_error(arr, nr):
    n = len(arr)
    res = 0
    for i in range(nr):
        val = 0
        for j in range(1, n + 1):
            if j & (2**i) == (2**i):
                val ^= int(arr[j - 1])
        res += val * (10**i)
    return int(str(res), 2)

def correct_error(arr, pos):
    if pos != 0:
        arr[pos - 1] = str(1 - int(arr[pos - 1]))  # Flip the bit at the error
position
    return arr

def remove_redundant_bits(arr, r):
    n = len(arr)
    res = ''
    j = 0
    for i in range(1, n + 1):
        if i != (2**j):  # Skip redundant bits
            res += arr[i - 1]
        else:
            j += 1
    return res

def binary_to_text(binary_data):
    text = ""
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i+8]  # Convert 8-bit binary to text
        text += chr(int(byte, 2))
    return text

def receiver():
    hamming_data = read_channel()
    nr = calculate_redundant_bits(len(hamming_data))
    error_pos = detect_error(hamming_data, nr)

    if error_pos == 0:
        print("No error detected.")
        correct_data = hamming_data
    else:
        print(f"Error detected at position: {error_pos}")
        correct_data = correct_error(list(hamming_data), error_pos)

    data_without_redundant_bits = remove_redundant_bits(correct_data, nr)
    ascii_text = binary_to_text(data_without_redundant_bits)

    print(f"Received text: {ascii_text}")

# Example usage
sender()
receiver()
```



```
Data sent and saved to Channel.txt: [1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0]
No error detected.
Received text: Harish
```