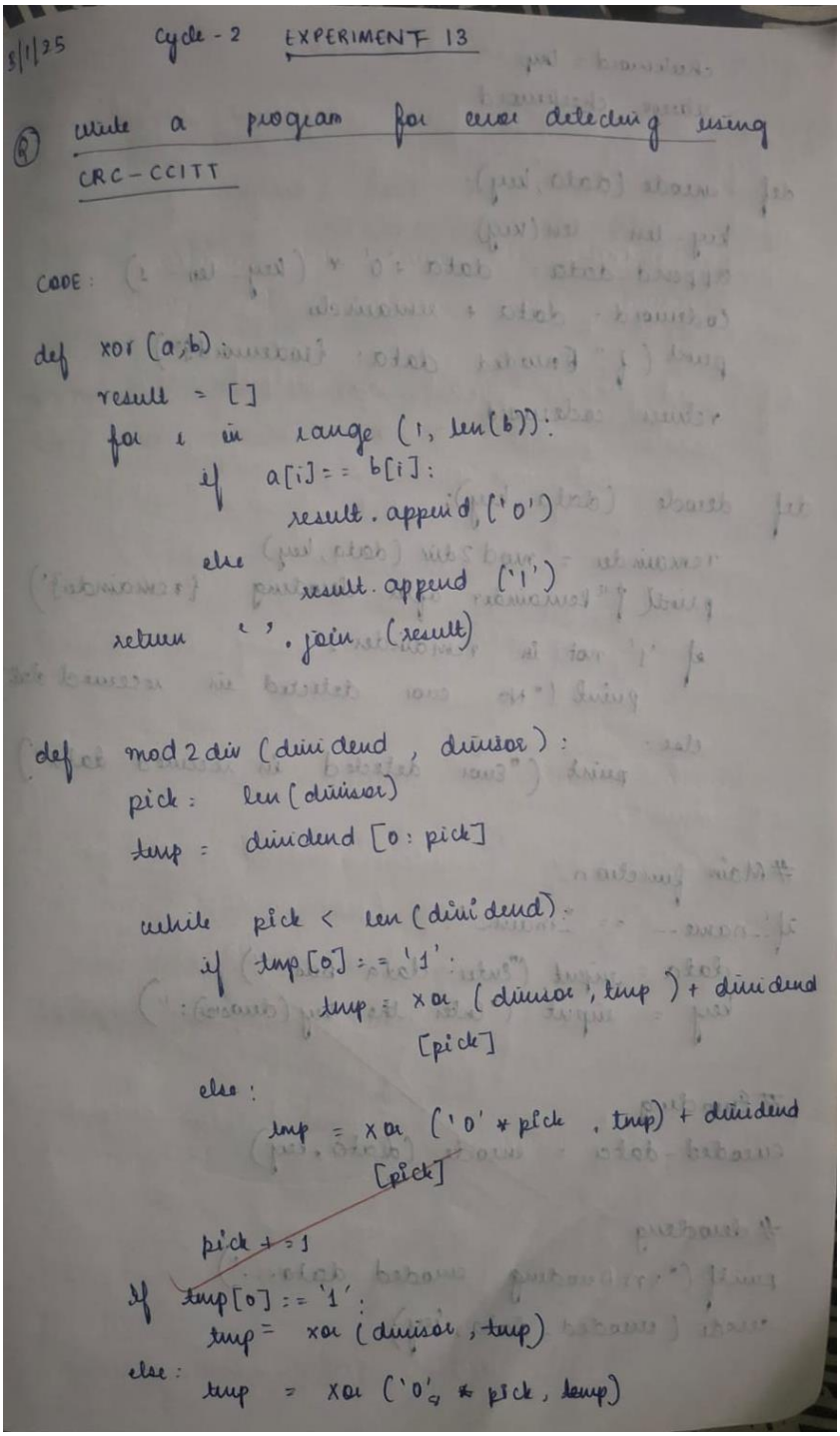
1|1|2025

Program 1:

Write a program for error detecting code using CRC-CCITT (16-bits)

Observation:



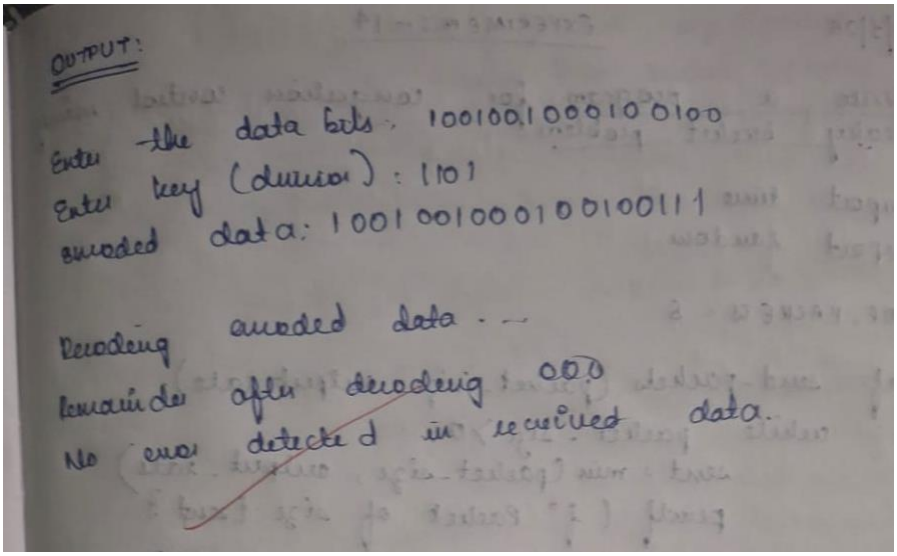
5/1/25 Cycle - 2 EXPERIMENT 13

(8) write a program for error detecting using CRC-CCITT

CODE:

```
def xor(a,b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else
            result.append('1')
    return ''.join(result)

def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0:pick]

    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend
                  [pick]

        else:
            tmp = xor('0'*pick, tmp) + dividend
                  [pick]

        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0'*pick, tmp)
```

```
    checkword = tmp
    return checkword

def encode(data, key):
    key_len = len(key)
    append_data = data + '0' * (key_len - 1)
    codeword = data + remainder
    print(f"Encoded data: {codeword}")
    return codeword

def decode(data, key):
    remainder = mod2div(data, key)
    print(f"Remainder after decoding: {remainder}")
    if '1' not in remainder:
        print("No error detected in received data")
    else:
        print("Error detected in received data")

#Main function
if __name__ == "__main__":
    data = input("Enter data bits: ")
    key = input("Enter the key (divisor): ")

#Encoding
encoded_data = encode(data, key)

#decoding
print("\nDecoding encoded data...")
decode(encoded_data, key)
```



Code:

```
def crc_ccitt_16_bitstream(bitstream: str, poly: int = 0x1021, init_crc: int =
0xFFFF) -> int:
    """
    Calculate the 16-bit CRC-CCITT checksum for a given binary string.
    """
    crc = init_crc
    for bit in bitstream:
        crc ^= int(bit) << 15  # Align the bit with CRC's uppermost bit
        for _ in range(8):  # Process each bit
            if crc & 0x8000:  # Check if the leftmost bit is set
                crc = (crc << 1) ^ poly
            else:
                crc <<= 1
            crc &= 0xFFFF  # Ensure CRC remains 16-bit
    return crc


def append_crc_to_bitstream(bitstream: str) -> str:
    """
    Append the calculated 16-bit CRC to the given bitstream.
    """
    crc = crc_ccitt_16_bitstream(bitstream)
    crc_bits = f"{crc:016b}"  # Convert CRC to a 16-bit binary string
    return bitstream + crc_bits


def verify_crc_bitstream(bitstream_with_crc: str) -> bool:
    """
    Verify the CRC of the given bitstream with CRC appended.
```

```
    """
    if len(bitstream_with_crc) < 16:
        return False  # Not enough bits to contain CRC
    data, received_crc = bitstream_with_crc[:-16], bitstream_with_crc[-16:]
    calculated_crc = crc_ccitt_16_bitstream(data)
    return calculated_crc == int(received_crc, 2)


# Main Program
if __name__ == "__main__":
    # User input for original bitstream
    message_bits = input("Enter the original bitstream (e.g., 11010011101100):
").strip()

    # Validate input
    if not all(bit in "01" for bit in message_bits):
        print("Invalid input. Please enter a binary bitstream (e.g.,
11010011101100).")
    else:
        # Calculate and append CRC
        bitstream_with_crc = append_crc_to_bitstream(message_bits)
        print(f"Transmitted bitstream with CRC: {bitstream_with_crc}")

        # User input for received bitstream
        user_bitstream = input("Enter the received bitstream for verification:
").strip()

        # Validate received input
        if not all(bit in "01" for bit in user_bitstream):
            print("Invalid input. Please enter a valid binary bitstream.")
        elif len(user_bitstream) < 16:
            print("Invalid input. Received bitstream must include at least 16
bits for CRC.")
        else:
            # Verify CRC
            is_valid = verify_crc_bitstream(user_bitstream)
            if is_valid:
                print("No errors detected. CRC valid.")
            else:
                print("Error detected! CRC invalid.")
```

Output:

```
PS C:\Users\HP\OneDrive\Desktop\cn lab> python crc-ccitt.py
Enter the original bitstream (e.g., 11010011101100): 11010011101100
Transmitted bitstream with CRC: 1101001110110010100110001001000
Enter the received bitstream for verification: 110100000000000101000001
Error detected! CRC invalid.
```