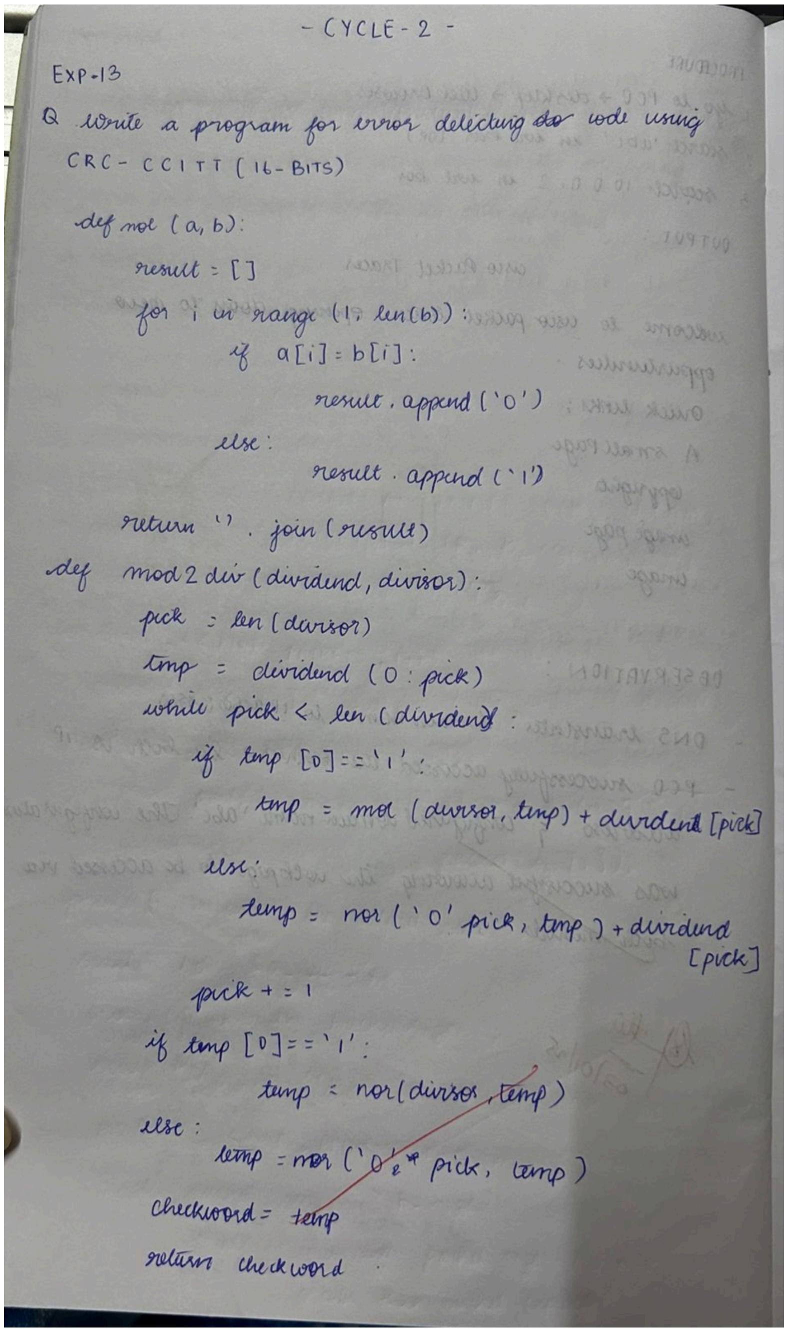
EXP-13

OBSERVATION

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

Observation:



- CYCLE - 2 -

EXP-13

Q Write a program for error detecting ~~do~~ code using CRC - CCITT (16-BITS)

```
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 mod2 div (dividend, divisor):
        pick = len (divisor)
        tmp = dividend (0 : pick)
        while pick < len (dividend):
                if tmp [0] = = '1':
                        tmp = xor (divisor, tmp) + dividend [pick]
                else:
                        temp = xor ('0' pick, tmp) + dividend
                                                        [pick]
                pick + = 1
        if tmp [0] = = '1':
                temp = xor(divisor, temp)
        else:
                temp = xor ('0'*pick, temp)
        checkword = temp
        return checkword
```

```
def encode (data, key):
        l - key = len(key) .
        appended_ data = data + '0' * ( len-key - 1)
        remainder = mod 2 div (appended - data. key)
        codeword = data + remainder .
        print ("Remainder :", remainder)
        print ('encoded data [data + remainder]", codeword)
        return codeword .

def decode-data (encoded - data, key)
        remainder = mod 2 div (encoded_data, key) .
        print ("Remainder after decoding : " , remainder)
        if '1' not in remainder:
            print ( "no error detected in received data")
        else
            print ("error detected in received data")

data = " 100100100010010 0 "
key = "1101 "
encoded - data = encode (data, key)
decoded _ data   = decode_data ( encoded - data, key )
```

OUTPUT :

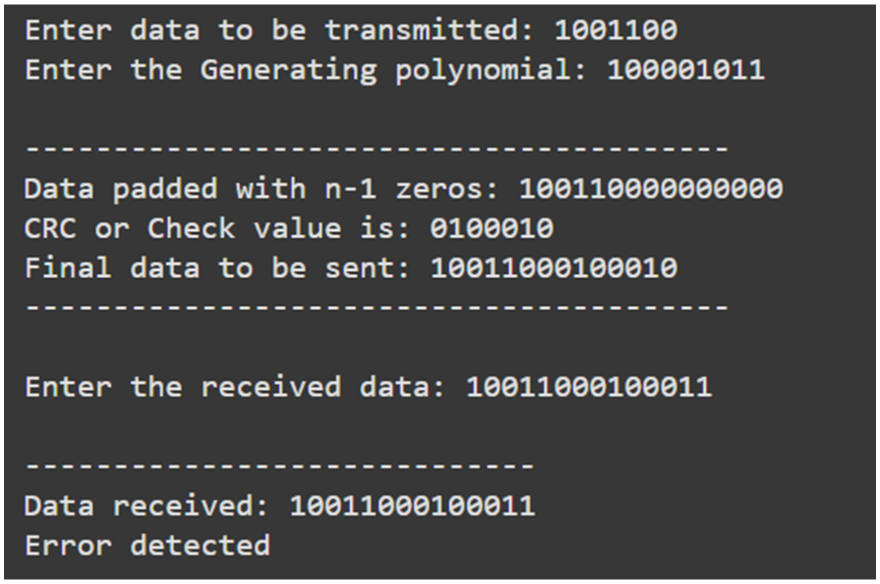
```
remainder = 11

encoded - data ( data + remainder ) = 100 1001 000100 10011

no error detected in received data .
```

[illegible]

Code:

```python
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:



```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------

Enter the received data: 10011000100011

-----------------------------
Data received: 10011000100011
Error detected
```