

## LABORATORY PROGRAM - 1

Write a program for error detecting code using CRC-CCITT (8-bits).

Cycle 2

Programs

(P1) CRC

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

Python Code:-

```
def crc_ccitt(data):  
    polynomial = 0x1021  
    initial_crc = 0xFFFF  
    crc = initial_crc  
    for byte in data.encode():  
        crc ^= (byte << 8)  
        for _ in range(8):  
            if crc & 0x8000:  
                crc = (crc << 1) ^ polynomial  
            else:  
                crc <<= 1  
        crc &= 0xFFFF  
    return crc
```

```
data = input('Enter data to calculate CRC-CCITT: ')  
print(f'checksum: {crc_ccitt(data):04x}')  
→ if MSB is 1, simulate division else move to next  
→ Marking: ensures CRC remains 16 bit
```

Output:-

Enter data to calculate CRC-CCITT: 1234  
CRC-CCITT checksum: 5349

Python Code:-

```
def crc_ccitt(data):  
    polynomial = 0x1021  
    initial_crc = 0xFFFF  
    crc = initial_crc  
    for byte in data.encode():  
        crc ^= (byte << 8)  
        for _ in range(8):  
            if crc & 0x8000:  
                crc = (crc << 1) ^ polynomial  
            else:  
                crc <<= 1  
        crc &= 0xFFFF  
    return crc
```

```
data = input('Enter data to calculate CRC-CCITT: ')  
print(f'checksum: {crc_ccitt(data): 04x}')  
→ if MSB is 1, simulate division else move to next
```

→ Marking: ensures CRC remains 16 bit

Output:-

Enter data to Calculate CRC-CCITT: 1234

CRC-CCITT checksum: 5349

```
return crc
data=input('Enter data to calculate CRC-CCITT: ')
print(f'checksum: {CRC-ccitt(data): 04x}')
→ if MSB is 1, simulate division else move to next
→ Marking: ensures CRC remains 16 bit
```

Output:-

Enter data to Calculate CRC-CCITT: 1234  
CRC-CCITT checksum: 5349

## Code

```
def xor(dividend, divisor):
    """Perform XOR operation between dividend and divisor."""
    result = ""
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result

def crc(data, gen_poly):
    """Compute the CRC check value using CRC-CCITT (8-bit)."""
    data_length = len(data)
    gen_length = len(gen_poly)

    # Append n-1 zeros to the data
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            # XOR operation if the first bit is 1
            check_value = xor(check_value, gen_poly)
        else:
            # Retain original check value if first bit is 0
            check_value = check_value[1:]

        # Shift left and add the next data bit
        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]

    return check_value[1:] # Remove the leading bit

def receiver(data, gen_poly):
    """Simulate the receiver side to check for errors."""
    print("\n-----")
    print("Data received:", data)

    # Perform CRC computation on received data
    remainder = crc(data, gen_poly)

    # Check if the remainder is all zeros
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")

if __name__ == "__main__":
    # Input data and generator polynomial
    data = input("Enter data to be transmitted: ")
    gen_poly = input("Enter the Generating polynomial: ")
```

```

# Compute CRC check value
check_value = crc(data, gen_poly)
print("\n-----")
print("Data padded with n-1 zeros:", data + '0' * (len(gen_poly) - 1))
print("CRC or Check value is:", check_value)

# Append check value to data for transmission
transmitted_data = data + check_value
print("Final data to be sent:", transmitted_data)
print("-----\n")

# Simulate the receiver side
received_data = input("Enter the received data: ")
receiver(received_data, gen_poly)

```

### 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

```