ASSIGNMENT – 2

CRC simulation

import crcmod.predefined

# Function to calculate CRC-32 checksum

def calculate\_crc32(data):

    crc32\_func = crcmod.predefined.mkCrcFun('crc-32')

    data\_bytes = data.encode('utf-8')

    checksum = crc32\_func(data\_bytes)

    return checksum

# Function to append CRC to the data

def append\_crc(data):

    checksum = calculate\_crc32(data)

    # Combine the original data with the CRC-32 checksum

    # The checksum is appended in a 4-byte format (4 bytes = 32 bits)

    data\_with\_crc = data + f"{checksum:08x}"

    return data\_with\_crc

# Function to verify the CRC-32 checksum at the receiver's end

def verify\_crc(received\_data):

    # Extract the original data and the CRC checksum

    original\_data = received\_data[:-8]  # Remove the last 8 characters (the checksum)

    received\_crc = int(received\_data[-8:], 16)  # Convert the last 8 characters to an integer

    # Calculate the CRC of the original data

    calculated\_crc = calculate\_crc32(original\_data)

    # Verify if the calculated CRC matches the received CRC

    if calculated\_crc == received\_crc:

        return True

    else:

        return False

# Sender side: prepare data and append CRC-32 checksum

data = "Hello,Ethernet!"

data\_with\_crc = append\_crc(data)

print(f"Original Data: {data}")

print(f"Data with CRC-32 appended: {data\_with\_crc}")

# Receiver side: verify the CRC checksum

is\_valid = verify\_crc(data\_with\_crc)

if is\_valid:

    print("CRC verification successful! Data is valid.")

else:

    print("CRC verification failed! Data is corrupted.")

# Introduce corruption in data

corrupted\_data\_with\_crc = data\_with\_crc[:5] + 'X' + data\_with\_crc[6:]  # Modify a character

print(f"Corrupted Data with CRC: {corrupted\_data\_with\_crc}")

# Verify the corrupted data

is\_valid\_corrupted = verify\_crc(corrupted\_data\_with\_crc)

if is\_valid\_corrupted:

    print("CRC verification successful! Data is valid.")

else:

    print("CRC verification failed! Data is corrupted.")

OUTPUT:

