**CRC-8 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)

padded\_data = data + '0' \* (gen\_length - 1)

check\_value = padded\_data[:gen\_length]

for i in range(data\_length):

if check\_value[0] == '1':

check\_value = xor(check\_value, gen\_poly)

else:

check\_value = check\_value[1:]

if i + gen\_length < len(padded\_data):

check\_value += padded\_data[i + gen\_length]

return check\_value[1:]

def receiver(data, gen\_poly):

"""Simulate the receiver side to check for errors."""

print("\n-----------------------------")

print("Data received:", data)

remainder = crc(data, gen\_poly)

if '1' in remainder:

print("Error detected")

else:

print("No error detected")

if \_\_name\_\_ == "\_\_main\_\_":

data = input("Enter data to be transmitted: ")

gen\_poly = input("Enter the Generating polynomial: ")

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)

transmitted\_data = data + check\_value

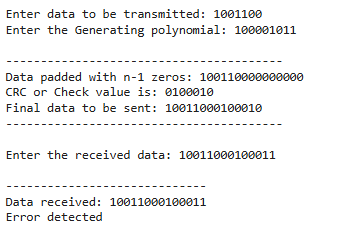
print("Final data to be sent:", transmitted\_data)

print("----------------------------------------\n")

received\_data = input("Enter the received data: ")

receiver(received\_data, gen\_poly)

**OUTPUT:**

****