**Experiment No. 4**

**Program Code:**

def xor(a, b):

# initialize result

result = []

# Traverse all bits, if bits are

# same, then XOR is 0, else 1

for i in range(1, len(b)):

if a[i] == b[i]:

result.append('0')

else:

result.append('1')

return ''.join(result)

# Performs Modulo-2 division

def mod2div(dividend, divisor):

# Number of bits to be XORed at a time.

pick = len(divisor)

# Slicing the dividend to appropriate

# length for particular step

tmp = dividend[0: pick]

while pick < len(dividend):

if tmp[0] == '1':

# replace the dividend by the result

# of XOR and pull 1 bit down

tmp = xor(divisor, tmp) + dividend[pick]

else: # If leftmost bit is '0'

# If the leftmost bit of the dividend (or the

# part used in each step) is 0, the step cannot

# use the regular divisor; we need to use an

# all-0s divisor.

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

# increment pick to move further

pick += 1

# For the last n bits, we have to carry it out

# normally as increased value of pick will cause

# Index Out of Bounds.

if tmp[0] == '1':

tmp = xor(divisor, tmp)

else:

tmp = xor('0'\*pick, tmp)

checkword = tmp

return checkword

# Function used at the sender side to encode

# data by appending remainder of modular division

# at the end of data.

def encodeData(data, key):

l\_key = len(key)

# Appends n-1 zeroes at end of data

appended\_data = data + '0'\*(l\_key-1)

remainder = mod2div(appended\_data, key)

# Append remainder in the original data

codeword = data + remainder

print("Remainder : ", remainder)

print("Encoded Data (Data + Remainder) : ",codeword)

# Input Data

data = input("Enter Binary Code:")

key = input("Enter Key:")

encodeData(data, key)

**Output:**

Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:/Users/ONKAR/AppData/Local/Programs/Python/Python312/CNS\_Ex\_4.py

Enter Binary Code:100100

Enter Key:1101

Remainder : 100

Encoded Data (Data + Remainder) : 100100100