**Name : Aarya Tiwari**

**Batch : B2:**

**Roll No. : 16010421119**

**Course:**

**Experiment No. :**

**Code:**

def majority(x, y, z):

*# Returns the majority bit*

    return (x & y) | (x & z) | (y & z)

def clock(register, majority\_bit):

    new\_bit = register[18] ^ majority\_bit

    register.pop()

    register.insert(0, new\_bit)

def generate\_keystream(key, num\_bits):

    R1 = list(key) + [0] \* (19 - len(key)) *# Makes the First Register 19 bits long*

    R2 = [0] \* 22 *# Makes the Second Register 22 bits long*

    R3 = [0] \* 23 *# Makes the Third Register 23 bits long*

    keystream = [] *# Stores the keystream*

    for \_ in range(num\_bits):

        majority\_bit = majority(R1[8], R2[10], R3[10])

        if R1[8] == majority\_bit:

            clock(R1, majority\_bit)

        if R2[10] == majority\_bit:

            clock(R2, majority\_bit)

        if R3[10] == majority\_bit:

            clock(R3, majority\_bit)

        keystream\_bit = R1[18] ^ R2[21] ^ R3[22]

        keystream.append(keystream\_bit)

    return keystream

def encrypt\_or\_decrypt(data, keystream):

    return [str(int(data[i]) ^ keystream[i]) for i in range(len(data))]

def a51\_encrypt(key, plaintext):

    keystream = generate\_keystream(key, len(plaintext))

    encrypted = encrypt\_or\_decrypt(plaintext, keystream)

    return ''.join(encrypted)

def a51\_decrypt(key, ciphertext):

*# Encryption and decryption are the same in a stream cipher*

    return a51\_encrypt(key, ciphertext)

def main():

    key = "1010101010101010101"  *# 19-bit key*

    plaintext = "1101101010101010101"  *# Example plaintext*

    ciphertext = a51\_encrypt(key, plaintext)

    decrypted\_text = a51\_decrypt(key, ciphertext)

    print("Plaintext:", plaintext)

    print("Ciphertext:", ciphertext)

    print("Decrypted Text:", decrypted\_text)

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

    main()

**Outcomes:**

**Conclusion:**