**key = 1010**

**pt = 1100**

**x = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**y = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**z = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**majority = 0**

**x = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**y = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**z = [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**k\_0 = 0**

**##############################################**

**x = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**y = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**z = [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**majority = 1**

**x = [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**y = [1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**z = [0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**k\_1 = 1**

**##############################################**

**x = [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**y = [1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**z = [0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**majority = 0**

**x = [0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**y = [1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**z = [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**k\_2 = 0**

**##############################################**

**x = [0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**y = [1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**z = [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**majority = 1**

**x = [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**y = [1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]**

**z = [0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]**

**k\_3 = 1**

**##############################################**

**key = 0101**

**pt = 1100**

**ct = 1001**

**class A51:**

**def \_\_init\_\_(self, key):**

**self.x = [int(b) for b in key[:19]]**

**self.y = [int(b) for b in key[19:41]]**

**self.z = [int(b) for b in key[41:]]**

**def majority(self, x, y, z):**

**return int(x + y + z > 1)**

**def step\_x(self):**

**t = self.x[13] ^ self.x[16] ^ self.x[17] ^ self.x[18]**

**self.x = [t] + self.x[:-1]**

**def step\_y(self):**

**t = self.y[20] ^ self.y[21]**

**self.y = [t] + self.y[:-1]**

**def step\_z(self):**

**t = self.z[7] ^ self.z[20] ^ self.z[21] ^ self.z[22]**

**self.z = [t] + self.z[:-1]**

**def generate\_keystream(self, length):**

**keystream = []**

**for \_ in range(length):**

**m = self.majority(self.x[8], self.y[10], self.z[10])**

**if self.x[8] == m:**

**self.step\_x()**

**if self.y[10] == m:**

**self.step\_y()**

**if self.z[10] == m:**

**self.step\_z()**

**keystream\_bit = self.x[18] ^ self.y[21] ^ self.z[22]**

**keystream.append(keystream\_bit)**

**return keystream**

**def xor\_bitsrings(a, b):**

**return [ai ^ bi for ai, bi in zip(a, b)]**

**def main():**

**key = input("Enter 64-bit key as a binary string: ")**

**if len(key) != 64 and not all (c in '01' for c in key):**

**print("Invalid key!")**

**return**

**plaintext = input("Enter the plaintext as a binary string: ")**

**if not all (c in '01' for c in plaintext):**

**print("Invalid plaintext!")**

**return**

**cipher = A51(key)**

**plaintext\_bits = [int (b) for b in plaintext]**

**keystream = cipher.generate\_keystream(len(plaintext\_bits))**

**ciphertext\_bits = xor\_bitsrings(plaintext\_bits, keystream)**

**ciphertext = ''.join(map(str, ciphertext\_bits))**

**print("Ciphertext (binary): ", ciphertext)**

**cipher = A51(key)**

**ciphertext\_bits = [int(b) for b in ciphertext]**

**decrypted\_bits = xor\_bitsrings(ciphertext\_bits, cipher.generate\_keystream(len(ciphertext\_bits)))**

**decrypted\_text = ''.join(map(str, decrypted\_bits))**

**print("Decrypted text (binary):", decrypted\_text)**

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

**main()**