# Импортируем модули для генерации случайными символов и для работы с системами счисления.  
import random  
import string  
# Определяем класс для работы с текстовым кодированием  
class TextEncoding:  
 @staticmethod # Статический метод, который не требует экземпляр класса для вызова  
 def determine\_alphabet(text):  
 # Определяем, используется ли латиница в шифре. Если используется, возвращаем латинский алфавит, в противном случае - кириллицу  
 if text[0] in string.ascii\_lowercase:  
 return string.ascii\_lowercase + string.digits  
 else:  
 return "абвгдеёжзийклмнопрстуфхцчшщъыьэюя" + string.digits  
 @staticmethod  
 def generate\_key(size, alphabet):  
 # Генерируем случайный ключ того же размера, что и вводимый текст  
 return "".join(random.choice(alphabet) for \_ in range(size))  
 @staticmethod  
 def to\_hex(coding):  
 # Конвертируем каждый символ в шестнадцатеричное представление и объединяем их все  
 return " ".join(hex(ord(character))[2:] for character in coding)  
 @staticmethod  
 def encode\_string(text, key):  
 # Возврат xor каждого символа в тексте с соответствующим символом в ключе  
 return "".join(chr(ord(char) ^ ord(key\_char)) for char, key\_char in zip(text, key))  
 @staticmethod  
 def xor\_texts(ciphertext1, ciphertext2):  
 # Возврат xor каждого символа в двух текстах  
 return "".join(chr(ord(char1) ^ ord(char2)) for char1, char2 in zip(ciphertext1, ciphertext2))  
# Получаем вводимые тексты  
plaintext1 = input("Введите первый открытый текст: ")  
plaintext2 = input("Введите второй открытый текст: ")  
  
# Определяем, какой алфавит использовать для генерации ключа  
alphabet = TextEncoding.determine\_alphabet(plaintext1)  
  
# Генерируем ключ  
key = TextEncoding.generate\_key(len(plaintext1), alphabet)  
  
# Выводим ключ и его шестнадцатеричное представление  
print(f"Ключ: {key}", f"Ключ в 16 бит: {TextEncoding.to\_hex(key)}", sep='\n')  
# Шифруем оба текста и выводим их и их шестнадцатеричные представления  
ciphertext1 = TextEncoding.encode\_string(plaintext1, key)  
ciphertext2 = TextEncoding.encode\_string(plaintext2, key)  
print(f"Первый зашифрованный текст: {ciphertext1}", f"Первый зашифрованный текст в 16 бит: {TextEncoding.to\_hex(ciphertext1)}", sep='\n')  
print(f"Второй зашифрованный текст: {ciphertext2}", f"Второй зашифрованный текст в 16 бит: {TextEncoding.to\_hex(ciphertext2)}", sep='\n')  
# Расшифровываем оба текста и выводим их  
decrypted\_text1 = TextEncoding.encode\_string(ciphertext1, key)  
decrypted\_text2 = TextEncoding.encode\_string(ciphertext2, key)  
print("Первый расшифрованный текст:", decrypted\_text1)  
print("Второй расшифрованный текст:", decrypted\_text2)  
  
# Выводим результат XOR между двумя зашифрованными текстами  
xor\_result = TextEncoding.xor\_texts(ciphertext1, ciphertext2)  
print("Результат XOR двух зашифрованных текстов:", xor\_result)





