Задание №1.

Зашифруйте сообщение *"this is exercise"*, используя один из следующих шифров. Игнорируйте пробелы между словами. Расшифруйте сообщение, чтобы получать первоначальный исходный текст.

- Аддитивный шифр с ключом = 20
- Мультипликативный шифр с ключом = 15
- Афинный шифр с ключом = (15, 20)

Решение

Возьмем афинный шифр с ключом (15, 20)

```
N = 26

def enc_afin(str, k1, k2):
    output = ""
    for char in str:
        #print(f"{char} {ord(char)}")
        output += chr(((ord(char) - ord("a")) * k1 + k2) % N +
ord("a"))
    return output


def gcdex(a, b, x, y):
    if (a == 0):
        return (b, 0, 1)
    d, x1, y1 = gcdex(b % a, a, 0, 0)
    x = y1 - (b // a) * x1
    y = x1
    return d, x, y

def inverse_mod(a, m):
    g, x, y = gcdex(a, m, 0, 0)
    if (g != 1):
        return None
    x = (x % m + m) % m
    return x

def dec_afin(str, k1, k2):
    output = ""
    for char in str:
        output += chr((((ord(char) - ord("a")) - k2) *
inverse_mod(k1, N)) % N + N) % N + ord("a"))
    return output

input = "thisisexercise"
k1 = 15
```

```
k2 = 20

print("Исходное сообщение:")
print(input)

print(f"Первый ключ: {k1}")
print(f"Второй ключ: {k2}")

encoded = enc_afin(input, k1, k2)
print("Сообщение, закодированное афинным шифром:")
print(encoded)

decoded = dec_afin(encoded, k1, k2)
print("Сообщение, декодированные афинным шифром:")
print(decoded)
```

```
Исходное сообщение:
thisisexercise
Первый ключ: 15
Второй ключ: 20
Сообщение, закодированное афинным шифром:
tvkekecbcpykec
Сообщение, декодированные афинным шифром:
thisisexercise
```

Задание №2

Зашифруйте сообщение *"the house is being sold tonight"*, используя один из следующих шифров. Игнорируйте пространство между словами. Расшифруйте сообщение, чтобы получить исходный текст.

- Шифр Виженера с ключом: *"dollars"*
- Шифр с автоматическим ключом = 7
- Шифр Плейфера с ключом, созданным в тексте (см. рис. 4.13)

Решение

Возьмем шифр Плейфера с ключом, созданным в тексте (см. рис. 4.13). В данном сообщении отсутствуют подряд идущие одинаковые буквы, а также длина сообщения 26, то есть четная. Поэтому дописывать символ для четности не надо.

```
def getCoords(char, key):
    for j in range(0, len(key)):
        for k in range(0, len(key[j])):
            if key[j][k].find(char.upper()) != -1:
                return (j, k)
    return (-1, -1)

def getChar(coord, key):
```

```
    return key[coord[0]][coord[1]][0].lower()

def enc_plfr(str, key):
    output = ""
    for i in range(0, len(str), 2):
        first = str[i]
        second = str[i+1]
        firstC = getCoords(first, key)
        secondC = getCoords(second, key)
        newFirstC=(firstC[0], secondC[1])
        newSecondC=(secondC[0], firstC[1])
        if firstC[0] == secondC[0]:
            newFirstC=(firstC[0], (firstC[1] + 1) % len(key[0]))
            newSecondC=(secondC[0], (secondC[1] + 1) % len(key[0]))
        elif firstC[1] == secondC[1]:
            newFirstC=((firstC[0] + 1) % len(key), firstC[1])
            newSecondC=((secondC[0] + 1) % len(key), secondC[1])

        newFirst = getChar(newFirstC, key)
        newSecond = getChar(newSecondC, key)
        output += newFirst + newSecond
    return output;


def dec_plfr(str, key):
    output = ""
    for i in range(0, len(str), 2):
        first = str[i]
        second = str[i+1]
        firstC = getCoords(first, key)
        secondC = getCoords(second, key)
        newFirstC=(firstC[0], secondC[1])
        newSecondC=(secondC[0], firstC[1])
        if firstC[0] == secondC[0]:
            newFirstC=(firstC[0], (firstC[1] - 1 + len(key[0])) %
len(key[0]))
            newSecondC=(secondC[0], (secondC[1] - 1 + len(key[0])) %
len(key[0]))
        elif firstC[1] == secondC[1]:
            newFirstC=((firstC[0] - 1 + len(key)) % len(key),
firstC[1])
            newSecondC=((secondC[0] - 1 + len(key)) % len(key),
secondC[1])
        newFirst = getChar(newFirstC, key)
        newSecond = getChar(newSecondC, key)
        output += newFirst + newSecond
    return output;

input = "thehouseisbeingsoldtonight"
key =   [["L", "G", "D", "B", "A"],
```

```
        ["Q", "M", "H", "E", "C"],
        ["U", "R", "N", "IJ", "F"],
        ["X", "V", "S", "O", "K"],
        ["Z", "Y", "W", "T", "P"]]

print("Исходное сообщение:")
print(input)

print("Ключ:")
print(key)

encoded = enc_plfr(input, key)
print("Сообщение, закодированное шифром Плейфера:")
print(encoded)

decoded = dec_plfr(encoded, key)
print("Сообщение, декодированное шифром Плейфера:")
print(decoded)
```

```
Исходное сообщение:
thehouseisbeingsoldtonight
Ключ:
[['L', 'G', 'D', 'B', 'A'], ['Q', 'M', 'H', 'E', 'C'], ['U', 'R', 'N',
'IJ', 'F'], ['X', 'V', 'S', 'O', 'K'], ['Z', 'Y', 'W', 'T', 'P']]
Сообщение, закодированное шифром Плейфера:
wecexiohnoeifidvxbbwsirbew
Сообщение, декодированное шифром Плейфера:
thehouseisbeingsoldtonight
```

Задание №3

Используйте шифр Виженера с ключевым словом *"HEALTH"*, чтобы зашифровать сообщение *"Life is full surprises" ("Жизнь полна сюрпризов")*.

Решение

```
def enc_vijn(str, key):
    output = ""
    for i in range(0, len(str)):
        output += chr( (ord(str[i]) - ord(key[i % len(key)]) - 2 *
ord('a')) % N + ord('a') )
    return output

def dec_vijn(str, key):
    output = ""
    for i in range(0, len(str)):
        output += chr( (ord(str[i]) + ord(key[i % len(key)]) + N) % N
+ ord('a') )
    return output

input = "lifeisfullofsurprises"
```

```
key = "health"

print("Исходное сообщение:")
print(input)

print("Ключ:")
print(key)

encoded = enc_vijn(input, key)
print("Сообщение, закодированное шифром Виженера:")
print(encoded)

decoded = dec_vijn(encoded, key)
print("Сообщение, декодированное шифром Виженера:")
print(decoded)
```

```
Исходное сообщение:
lifeisfullofsurprises
Ключ:
health
Сообщение, закодированное шифром Виженера:
ssthdzmezojmzefsmpzog
Сообщение, декодированное шифром Виженера:
lifeisfullofsurprises
```

Задание №4

Используйте шифр Плейфера, чтобы зашифровать сообщение *"The key hidden under the door pad" ("ключ спрятан под ковриком у двери")*. Ключ засекречивания можно составить, заполняя первую и вторую часть строки со словом *"GUIDANCE"* и заполняя остальную часть матрицы с остальной частью алфавита.

Решение

В сообщении присутствуют одинаковые подряд идущие буквы *("d" и "o")*, которые нужно разделить другим символом, например, *"x"*. Плюс длина сообщения нечетная, поэтому добавим *"x"* и в конец сообщения.

Напишем функцию обработки строки, а так же составим ключ по строке.

```
def pred_plfr(str, symb):
    output = ""
    for i in range(0, len(str) - 1):
        output += str[i]
        if str[i] == str[i + 1]:
            output += symb
    output += str[-1]
    if len(output) % 2 == 1:
        output += symb
```

```
    return output

key =   [["G", "U", "IJ", "D", "A"],
         ["N", "C", "E", "B", "F"],
         ["H", "K", "L", "M", "O"],
         ["P", "Q", "R", "S", "T"],
         ["V", "W", "X", "Y", "Z"]]

input = "thekeyishiddenunderthedoorpad"
corr_input = pred_plfr(input, "x")

print("Исходное сообщение:")
print(input)

print("Скорректированное сообщение:")
print(corr_input)

print("Ключ:")
print(key)

encoded = enc_plfr(corr_input, key)
print("Сообщение, закодированное шифром Плейфера:")
print(encoded)

decoded = dec_plfr(encoded, key)
print("Сообщение, декодированное шифром Плейфера:")
print(decoded)
```

```
Исходное сообщение:
thekeyishiddenunderthedoorpad
Скорректированное сообщение:
thekeyishidxdenunderthedoxorpadx
Ключ:
[['G', 'U', 'IJ', 'D', 'A'], ['N', 'C', 'E', 'B', 'F'], ['H', 'K',
'L', 'M', 'O'], ['P', 'Q', 'R', 'S', 'T'], ['V', 'W', 'X', 'Y', 'Z']]
Сообщение, закодированное шифром Плейфера:
poclbxdrlgiyibcgbglxpobilzlttgiy
Сообщение, декодированное шифром Плейфера:
thekeyishidxdenunderthedoxorpadx
```

Задание №5

Используйте шифр Хилла, чтобы зашифровать сообщение *"We live in an insecure world" ("Мы живем в опасном мире")*. Применять следующий ключ:

K = |3 2|

.....|5 7|

Решение

Подрубаем numpy, чтобы упростить взаимодействие с матрицами.

```
import numpy as np
```

Для упрощения жизни не будем заниматься поиском обратной по модулю матрицы, ибо это не самая простая задача. Учитывая это, нам потребуется всего один метод, так как по сути в случае кодирования мы умножаем ключ на матрицу с исходным сообщением, а в случае декодирования мы умножаем "обратный" ключ на закодированное сообщение.

```
def hill(str, key):
    l = key.shape[0]
    # размер ключа
    if len(str) % l != 0:
        str += "x" * (l - len(str) % l)
    # дописываем "x" в конец сообщения, чтобы оно преобразовывалось в
прямоугольную матрицу с высотой l
    str_arr = np.asarray([ord(char) - ord('a') for char in str])
    # перегоняем строку в np.array с кодами символов
    str_arr.shape = (int(len(str) / l), l)
    str_arr = str_arr.transpose()
    # натягиваем вектор на матрицу и транспонируем её, получая нужную
для умножения матрицу
    multed = np.matmul(key, str_arr) % N
    # перемножаем ключ и сообщение по модулю N
    multed = multed.transpose().ravel()
    # назад транспонируем и схлопываем в вектор нашу матрицу, чтобы
получить строку
    output = "".join([chr(num + ord('a')) for num in multed.tolist()])
    # перегоняем числа в буквы и собираем из них строку
    return output


input = "weliveinaninsecureworld"

key = np.array([[3, 2],
                [5, 7]])

inv_key = np.array([[3, 14],
                    [9, 5]])

print("Исходное сообщение:")
print(input)

print("Ключ:")
print(key)

print("\"Обратный\" ключ:")
```

```
print(inv_key)

encoded = hill(input, key)
print("Сообщение, закодированное шифром Хилла:")
print(encoded)

decoded = hill(encoded, inv_key)
print("Сообщение, декодированное шифром Хилла:")
print(decoded)
```

```
Исходное сообщение:
weliveinaninsecureworld
Ключ:
[[3 2]
 [5 7]]
"Обратный" ключ:
[[ 3 14]
 [ 9  5]]
Сообщение, закодированное шифром Хилла:
wixhtdybanybkouuhjqavgdu
Сообщение, декодированное шифром Хилла:
weliveinaninsecureworldx
```