МАТЕРИАЛЫ ЗАДАНИЙ МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО ПО ПРОФИЛЮ «КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ»

2025/2026 УЧЕБНЫЙ ГОД

ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП 8-10 КЛАССЫ

СОДЕРЖАНИЕ

Задача 1. RGB

Вариант 1

Отдел информационной безопасности «Тринити-Код» расследует утечку конфиденциальных данных. В ходе расследования у системного администратора обнаружено странное изображение. Известно, что администратор скрыл в нем имя своего сообщника. Вам необходимо проанализировать изображение и расшифровать имя сообщника, с которым администратор строил свои планы.

В качестве ответа укажите имя сообщника.

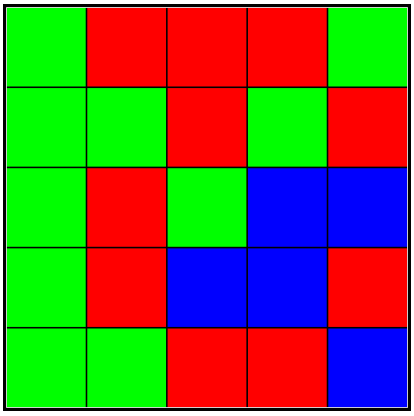

Рисунок – Странное изображение

Решение

Имеется картинка 5*5, состоящие из трех цветов – красного, синего и зеленого.

На первый взгляд ничего приметного в этом нет. Однако, от нас требуется в задаче получить имя злоумышленника, а значит тут скрыто осмысленное слово.

Исходя из вводных данных о количестве квадратов, количестве цветов и что ответ – осмысленное слово, предположим, что цвета – это троичная система счисления, где R – 0, G – 1, B – 2.

Получается следующая таблица:

1	0	0	0	1
1	1	0	1	0
1	0	1	2	2
1	0	2	2	0
1	1	0	0	2

Осмысленного сообщения не получилось, значит это ещё не все. Попробуем перевести из троичной системы в двоичную и затем в ASCII.

10001	1010010	R
11010	1101111	o
10122	1100010	b
10220	1101001	i
11002	1101110	n

Ответ: Robin

Вариант 2

Отдел информационной безопасности «Тринити-Код» расследует утечку конфиденциальных данных. В ходе расследования у системного администратора обнаружено странное изображение. Известно, что администратор скрыл в нем имя своего сообщника. Вам необходимо проанализировать изображение и расшифровать имя сообщника, с которым администратор строил свои планы.

В качестве ответа укажите имя сообщника.

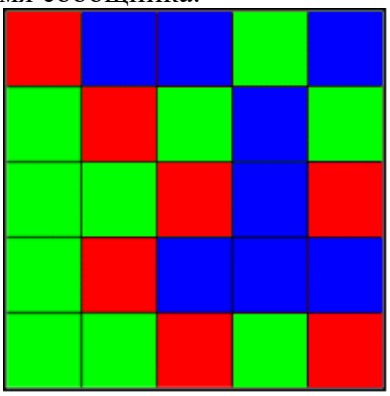

Рисунок – Странное изображение

Решение

Имеется картинка 5*5, состоящие из трех цветов – красного, синего и зеленого.

На первый взгляд ничего приметного в этом нет. Однако, от нас требуется в задаче получить имя злоумышленника, а значит тут скрыто осмысленное слово.

Исходя из вводных данных о количестве квадратов, количестве цветов и что ответ – осмысленное слово, предположим, что цвета – это троичная система счисления, где R – 0, G – 1, B – 2.

Получается следующая таблица:

0	2	2	1	2
1	0	1	2	1
1	1	0	2	0
1	0	2	2	2
1	1	0	1	0

Осмысленного сообщения не получилось, значит это ещё не все. Попробуем перевести из троичной системы в двоичную и затем в ASCII.

02212	1001101	M
10121	1100001	a
11020	1110010	r
10222	1101011	k
11010	1101111	o

Ответ: Marko

Задача 2. Хэширование

Вариант 1

Сотрудник отдела информационной безопасности компании «АС-Крипто» на рабочем столе системного администратора обнаружил ссылку на git-репозиторий. В репозитории содержится:

- описание алгоритма хеширования;
- исходный код функции хеширования;
- файл test_results.txt.

Из описания стало известно, что алгоритм хэширования используется администратором для хранения паролей. Алгоритм предполагает использование «соли» – 4-символьная строка, состоящая из заглавных и строчных латинских букв (A-Z, a–z), добавляемая к хешируемой строке.

Исходный код функций хеширования представлен в листинге.

Листинг 1. Исходный код функций хеширования

Python

```
# Нормализует соль до ровно 4 символов
def normalize_salt(salt: str) -> str:
    # Если соль пустая - возвращаем пустую строку
    if len(salt) == 0:
        return ''
    if len(salt) < 4:
        # Если соль короче 4 символов - повторяем её по кругу
        return ''.join(salt[i % len(salt)] for i in range(4))
    else:
        # Если длиннее 4 - берём только первые 4 символа
        return salt[:4]

# Добавляет соль в пароль
def weave_salt(password: str, salt: str) -> str:
    salt = normalize_salt(salt)
    half = len(password) // 2  # Находим середину пароля (целочисленное деление)
    return password[:half] + salt + password[half:] + salt

# Вычисляет хеш строки
def custom_hash(s: str) -> str:
    # Начальные значения - 8 магических байт (основа хеша)
    h = [0xEF, 0xAC, 0xEB, 0xCA, 0xAD, 0xDF, 0xBE, 0xDA]
    for c in s:
        # Для каждого элемента h[i] умножаем его на 65537,
        # затем на (i+1)
        # и добавляем ASCII-код символа c
        h = [ (h[i] * 65537 * (i+1) + ord(c)) % (2**16) for i in range(len(h))]
    # Преобразуем число в HEX-формат и конвертируем в строку
    hash_str = ''.join(f"{x:04x}" for x in h)
    # Берём последние 4 символа исходной строки
    hash_4 = s[-4:]
    # Переводим каждый символ в 2-значный HEX-код
    salt = ''.join(f"{ord(ch):02x}" for ch in hash_4)
    # Модифицируем последние 8 символов хеша
    return hash_str[:-8] + salt

PASSWORD = ""
REAL_SALT = ""
woven = weave_salt(PASSWORD, REAL_SALT)
hash_val = custom_hash(woven)
print(f"Сгенерированный хеш: {hash_val}")
```

C++

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
using namespace std;

// Функция normalize_salt — нормализует соль до 4 символов
string normalize_salt(string salt) {
    // Если соль пустая — возвращаем пустую строку
    if (salt.length() == 0) {
        return "";
    }
    // Если соль короче 4 символов — повторяем её символы по кругу
    if (salt.length() < 4) {
        string result = "";
        for (int i = 0; i < 4; i++) {
            result += salt[i % salt.length()];
        }
        return result;
    }
    else {
        // Берём подстроку: начиная с позиции 0, длиной 4 символа
        return salt.substr(0, 4);
    }
}

// Функция weave_salt — добавляет соль в пароль
string weave_salt(string password, string salt) {
    salt = normalize_salt(salt);
    int half = password.length() / 2;
    string first_part = password.substr(0, half);     // первая половина пароля
    string second_part = password.substr(half);       // вторая половина пароля
    return first_part + salt + second_part + salt;
}

// Функция custom_hash — вычисляет хеш строки
string custom_hash(string s) {
    // Начальные значения - 8 магических байт (основа хеша)
    unsigned int h[8] = { 0xEF, 0xAC, 0xEB, 0xCA, 0xAD, 0xDF, 0xBE, 0xDA };
    for (char c : s) {
        // Для каждого элемента h[i] умножаем его на 65537,
        // затем на (i + 1)
        // и добавляем ASCII-код символа c
        for (int i = 0; i < 8; i++) {
            h[i] = (h[i] * 65537*(i+1) + (unsigned int)c) % 65536;
        }
    }
    // Собираем строку из 8 * 4 = 32 hex-символов
    // Каждое число форматируется как 4 hex-цифры
    stringstream hash_stream;
    for (int i = 0; i < 8; i++) {
        hash_stream << hex << setw(4) << setfill('0') << h[i];
    }
    string hash_str = hash_stream.str(); // итого 32 символа
    // Берём последние 4 символа исходной строки
    string hash_4 = s.substr(s.length() >= 4 ? s.length() - 4 : 0);
    // Переводим каждый символ в 2-значный HEX-код
    string salt_hex = "";
    for (char ch : hash_4) {
        stringstream ss;
        ss << hex << setw(2) << setfill('0') << (int)(unsigned char)ch;
```

```
        salt_hex += ss.str();
    }
    // Модифицируем последние 8 символов хеша
    string final_hash = hash_str.substr(0, hash_str.length() - 8) + salt_hex;
    return final_hash;
}

// Главная функция — точка входа программы
int main() {
    setlocale(LC_ALL, "ru");
    string PASSWORD = "";
    string REAL_SALT = "";
    string woven = weave_salt(PASSWORD, REAL_SALT);
    string hash_val = custom_hash(woven);
    cout << "Сгенерированный хеш: " << hash_val << endl;
    return 0;
}
```

В файле *test_results.txt* содержатся следующие строки:

```
password
TEST
06e2ee6e82daf6f0d630d21a54455354
```

Определите, какая фраза использовалась для подмешивания («соль»), если известно, что пароль – `ikbmtuci`, а его хэш-значение – `06955dcf1ae16061178b4b9b56494345`.

Решение

Имеется алгоритм хэширования, из его анализа видно, что «соль» встраивается в середине и в конце пароля:

```
return password[:half] + salt + password[half:] + salt
```

Далее проанализируем саму функцию хеширования (`custom_hash()`), в которой имеется уязвимость: в результат хэширования, в конце строки 8 символов хэша заменяется на hex-значение «соли»:

```
        hash_4 = s[-4:]
        salt = ''.join(f"{ord(ch):02x}" for ch in hash_4)
        return hash_str[:-8] + salt
```

Нам известны тестовые значения и можно проверить эту гипотезу взяв последние 8 символов тестового хэша и перевести их в ASCII:

```
54 - T
45 - E
53 - S
54 - T
```

Совпадает, значит для хэш-значения –

```
3b9cbee0861c001bbc66bdb8155e58573baa6b3620f75f7b6597134b56494345
```

соль будет:

```
56 - V
49 - I
43 - C
45 - E
```

Ответ: VICE

Вариант 2

Сотрудник отдела информационной безопасности компании «АС-Крипто» на рабочем столе системного администратора обнаружил ссылку на git-репозиторий. В репозитории содержится:

- описание алгоритма хеширования;
- исходный код функции хеширования;
- файл test_results.txt.

Из описания стало известно, что алгоритм хэширования используется администратором для хранения паролей. Алгоритм предполагает использование «соли» – 4-символьная строка, состоящая из заглавных и строчных латинских букв (A-Z, a–z), добавляемая к хешируемой строке.

Исходный код функций хеширования представлен в листинге.

Листинг 1. Исходный код функций хеширования

Python

```
# Нормализует соль до ровно 4 символов
def normalize_salt(salt: str) -> str:
    # Если соль пустая - возвращаем пустую строку
    if len(salt) == 0:
        return ''
    if len(salt) < 4:
        # Если соль короче 4 символов - повторяем её по кругу
        return ''.join(salt[i % len(salt)] for i in range(4))
    else:
        # Если длиннее 4 - берём только первые 4 символа
        return salt[:4]

# Добавляет соль в пароль
def weave_salt(password: str, salt: str) -> str:
    salt = normalize_salt(salt)
    half = len(password) // 2  # Находим середину пароля (целочисленное деление)
    return password[:half] + salt + password[half:] + salt

# Вычисляет хеш строки
def custom_hash(s: str) -> str:
    # Начальные значения - 8 магических байт (основа хеша)
    h = [0xEF, 0xAC, 0xEB, 0xCA, 0xAD, 0xDF, 0xBE, 0xDA]
    for c in s:
        # Для каждого элемента h[i] умножаем его на 65537,
        # затем на (i+1)
        # и добавляем ASCII-код символа c
        h = [ (h[i] * 65537 * (i+1) + ord(c)) % (2**16) for i in range(len(h))]
    # Преобразуем число в HEX-формат и конвертируем в строку
    hash_str = ''.join(f"{x:04x}" for x in h)
    # Берём последние 4 символа исходной строки
    hash_4 = s[-4:]
    # Переводим каждый символ в 2-значный HEX-код
    salt = ''.join(f"{ord(ch):02x}" for ch in hash_4)
    # Модифицируем последние 8 символов хеша
    return hash_str[:-8] + salt

PASSWORD = ""
REAL_SALT = ""
woven = weave_salt(PASSWORD, REAL_SALT)
hash_val = custom_hash(woven)
print(f"Сгенерированный хеш: {hash_val}")
```

C++

```
#include <iostream>
#include <string>
#include <sstream>
#include <iomanip>
using namespace std;

// Функция normalize_salt — нормализует соль до 4 символов
string normalize_salt(string salt) {
    // Если соль пустая — возвращаем пустую строку
    if (salt.length() == 0) {
        return "";
    }
    // Если соль короче 4 символов — повторяем её символы по кругу
    if (salt.length() < 4) {
        string result = "";
        for (int i = 0; i < 4; i++) {
            result += salt[i % salt.length()];
        }
        return result;
    }
    else {
        // Берём подстроку: начиная с позиции 0, длиной 4 символа
        return salt.substr(0, 4);
    }
}

// Функция weave_salt — добавляет соль в пароль
string weave_salt(string password, string salt) {
    salt = normalize_salt(salt);
    int half = password.length() / 2;
    string first_part = password.substr(0, half);     // первая половина пароля
    string second_part = password.substr(half);       // вторая половина пароля
    return first_part + salt + second_part + salt;
}

// Функция custom_hash — вычисляет хеш строки
string custom_hash(string s) {
    // Начальные значения - 8 магических байт (основа хеша)
    unsigned int h[8] = { 0xEF, 0xAC, 0xEB, 0xCA, 0xAD, 0xDF, 0xBE, 0xDA };
    for (char c : s) {
        // Для каждого элемента h[i] умножаем его на 65537,
        // затем на (i + 1)
        // и добавляем ASCII-код символа c
        for (int i = 0; i < 8; i++) {
            h[i] = (h[i] * 65537*(i+1) + (unsigned int)c) % 65536;
        }
    }
    // Собираем строку из 8 * 4 = 32 hex-символов
    // Каждое число форматируется как 4 hex-цифры
    stringstream hash_stream;
    for (int i = 0; i < 8; i++) {
        hash_stream << hex << setw(4) << setfill('0') << h[i];
    }
    string hash_str = hash_stream.str(); // итого 32 символа
    // Берём последние 4 символа исходной строки
    string hash_4 = s.substr(s.length() >= 4 ? s.length() - 4 : 0);
    // Переводим каждый символ в 2-значный HEX-код
    string salt_hex = "";
    for (char ch : hash_4) {
        stringstream ss;
        ss << hex << setw(2) << setfill('0') << (int)(unsigned char)ch;
```

```
        salt_hex += ss.str();
    }
    // Модифицируем последние 8 символов хеша
    string final_hash = hash_str.substr(0, hash_str.length() - 8) + salt_hex;
    return final_hash;
}

// Главная функция — точка входа программы
int main() {
    setlocale(LC_ALL, "ru");
    string PASSWORD = "";
    string REAL_SALT = "";
    string woven = weave_salt(PASSWORD, REAL_SALT);
    string hash_val = custom_hash(woven);
    cout << "Сгенерированный хеш: " << hash_val << endl;
    return 0;
}
```

В файле *test_results.txt* содержатся следующие строки:

```
password
TEST
06e2ee6e82daf6f0d630d21a54455354
```

Определите, какая фраза использовалась для подмешивания («соль»), если известно, что пароль – `mtuciikb`, а его хэш-значение – `06ad68f2fd65f9d067dbe05659414d4c`.

Решение

Имеется алгоритм хэширования, из его анализа видно, что «соль» встраивается в середине и в конце пароля:

```
return password[:half] + salt + password[half:] + salt
```

Далее проанализируем саму функцию хеширования (`custom_hash()`), в которой имеется уязвимость: в результат хэширования, в конце строки 8 символов хэша заменяется на hex-значение «соли»:

```
        hash_4 = s[-4:]
        salt = ''.join(f"{ord(ch):02x}" for ch in hash_4)
        return hash_str[:-8] + salt
```

Нам известны тестовые значения и можно проверить эту гипотезу взяв последние 8 символов тестового хэша и перевести их в ASCII:

```
54 – T
45 – E
53 – S
54 – T
```

Совпадает, значит для хэш-значения –

`3ba0bf2588510b57e3dc2294d6cd738e7afe80ced9b1bbcb86e61b4d59414d4c`

соль будет:

```
59 – Y
41 – A
4d – M
4c – L
```

Ответ: YAML

Задача 3. Забывчивый сотрудник

Вариант 1

Один из сотрудников компании «Redis corp» забыл пароль для доступа к личному облачному хранилищу. Он точно помнит, что длина пароля составляет **6 символов**, и он содержит только латинские буквы и цифры (`0-9a-z`). Поняв, что не сможет вспомнить свой пароль, он решил написать программу перебора всех возможных паролей последовательно, начиная с «`000000`», «`000001`», …, и заканчивая «`zzzzzz`». Программа на вход принимает начальный пароль и последовательно перебирает все последующие, используя следующий алфавит символов:

№	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Символ	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**
№	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Символ	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**

Сотрудник запустил программу перебора в **10:00**. На проверку каждого пароля требуется **0,02 мс**.

Коллега по работе знал забытый пароль, и сказал, что он не успеет подобрать пароль до конца рабочего дня (**до 18:00**).

Определите, с какого пароля необходимо начинать последовательный перебор, не меняя при этом саму программу, чтобы гарантированно подобрать пароль до конца рабочего дня.

Решение

Шаг 1. Определим алфавит и общее количество паролей.

Символы: 0-9 (10 цифр) + a-z (26 букв) = 36 символов. Длина пароля – 6 символов. Всего паролей: $36^6 = 2\ 176\ 782\ 336$.

Шаг 2. Определим, сколько паролей можно проверить за рабочий день

Рабочее время: с 10:00 до 18:00 = 8 часов или 8 * 3 600 000 мс = 28 800 000 мс.

Паролей в мс: 1 / 0,02 = 50. Итого можно проверить: 28 800 000 * 50 = 1 440 000 000 паролей за полный рабочий день.

Шаг 3. Поиск стартового пароля

Коллега сказал, что начиная с “000000” пароль найти не успеют – значит пароль находится за пределами первых 1 440 000 000 позиций.

Чтобы гарантированно найти пароль, нужно начать перебор с такой позиции, чтобы от неё до “zzzzzz” оставалось не более 1 440 000 000 паролей:

Стартовая позиция = 2 176 782 336 − 1 440 000 000 = 736 782 336.

Осталось посчитать, какой пароль соответствует порядковому номеру 736 782 336. Для этого напишем программу на языке программирования Python.

Листинг программы

```
# Константы
ALPHABET = "0123456789abcdefghijklmnopqrstuvwxyz"  # 36 символов
BASE = len(ALPHABET)          # основание системы счисления = 36
PASSWORD_LEN = 6              # длина пароля

WORK_HOURS = 8                # рабочих часов (10:00-18:00)
MS_PER_HOUR = 3_600_000       # миллисекунд в часе
TIME_PER_CHECK_MS = 0.02      # время проверки одного пароля, мс

# Шаг 1: Общее количество паролей
total_passwords = BASE ** PASSWORD_LEN
print(f"Всего паролей: {total_passwords}")
```

```
# Шаг 2: Сколько паролей можно проверить за рабочий день
total_ms = WORK_HOURS * MS_PER_HOUR          # всего миллисекунд
checks_per_ms = 1 / TIME_PER_CHECK_MS        # паролей в мс
max_checks = int(total_ms * checks_per_ms)   # максимум паролей за смену
print(f"Можно проверить за рабочий день: {max_checks}")

# Шаг 3: Стартовая позиция (индекс в общем списке паролей)
start_index = total_passwords - max_checks
print(f"Стартовая позиция (индекс): {start_index}")

# Шаг 4: Перевод индекса в пароль (десятичное → 36-ричное)
def index_to_password(index, length, alphabet):
    password = []
    for _ in range(length):
        password.append(alphabet[index % len(alphabet)])  # остаток = текущий символ
        index //= len(alphabet)                           # целочисленное деление
    return ''.join(reversed(password))               # разворачиваем (шли с конца)

start_password = index_to_password(start_index, PASSWORD_LEN, ALPHABET)
print(f"\nНачинать перебор с пароля: '{start_password}'")
```

Результат работы программы:

```
Всего паролей: 36^6 = 2,176,782,336
Можно проверить за смену: 1,440,000,000
Стартовая позиция (индекс): 736,782,336

Начинать перебор с пароля: 'c6nsw0'
```

Ответ: c6nsw0 или 736 782 336

Вариант 2

Один из сотрудников компании «Redis corp» забыл пароль для доступа к личному облачному хранилищу. Он точно помнит, что длина пароля составляет **6 символов**, и он содержит только латинские буквы и цифры (`0-9a-z`). Поняв, что не сможет вспомнить свой пароль, он решил написать программу перебора всех возможных паролей последовательно, начиная с «`000000`», «`000001`», …, и заканчивая «`zzzzzz`». Программа на вход принимает начальный пароль и последовательно перебирает все последующие, используя следующий алфавит символов:

№	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Символ	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**
№	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Символ	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**

Сотрудник хочет запустить программу после окончания рабочего дня в **19:00**. На проверку каждого пароля требуется **0,04 мс**.

Коллега по работе знал забытый пароль, и сказал, что он не успеет подобрать пароль до начала следующего рабочего дня (**до 10:00**).

Определите, с какого пароля необходимо начинать последовательный перебор, не меняя при этом саму программу, чтобы гарантированно подобрать пароль к началу следующего рабочего дня.

Решение

Шаг 1. Определим алфавит и общее количество паролей.

Символы: 0-9 (10 цифр) + a-z (26 букв) = 36 символов. Длина пароля – 6 символов. Всего паролей: $36^6 = 2\ 176\ 782\ 336$.

Шаг 2. Определим, сколько паролей можно проверить за указанный период

Время работы программы – с 19:00 до 10:00 = 15 часов или $15 * 3\ 600\ 000 = 54\ 000\ 000$ мс.

Скорость работы – $1 \div 0{,}04 = 25$ паролей в мс. Таком образом, всего паролей может быть обработано:

$$54\ 000\ 000 * 25 = 1\ 350\ 000\ 000 \text{ паролей.}$$

Шаг 3. Поиск стартового пароля

Коллега сказал, что начиная с “000000” пароль найти не успеют – значит пароль находится за пределами первых 1 350 000 000 позиций.

Чтобы гарантированно добраться до “zzzzzz”, нужно начать не позднее $2\ 176\ 782\ 336 - 1\ 350\ 000\ 000 = 826\ 782\ 336$ пароля.

Осталось посчитать, какой пароль соответствует порядковому номеру 826 782 336.

Шаг 4. Определение комбинации пароля

Необходимо перевести число 826 782 336 из десятичной в 36-ричную систему в соответствии с индексом каждого символа (см. таблицу из задания).

Последовательно делим число на 36:

Шаг	**Делимое**	**Результат**	**Остаток**	**Символ**
1	826 782 336	22 966 176	0	**0**
2	22 966 176	637 949	12	**c**
3	637 949	17 720	29	**t**
4	17 720	492	8	**8**
5	492	13	24	**o**
6	13	0	13	**d**

Переводим (читаем) остатки *снизу вверх:* d, o, 8, t, c, 0.

Ответ: do8tc0 или 826 782 336

Задача 4. Ping

Вариант 1

Отдел информационной безопасности компании «АС-Крипто» расследует деятельность внутреннего злоумышленника, который, по данным разведки, планирует встретиться со своим сообщником для передачи конфиденциальных данных. Команда аналитиков смогла перехватить часть трафика, где передавалось место встречи злоумышленников, однако понять, как именно оно передавалось им не удалось.

На основе имеющегося трафика, определите передаваемое место встречи и внутреннего злоумышленника.

В качестве ответа укажите место встречи и IP-адрес внутреннего злоумышленника.

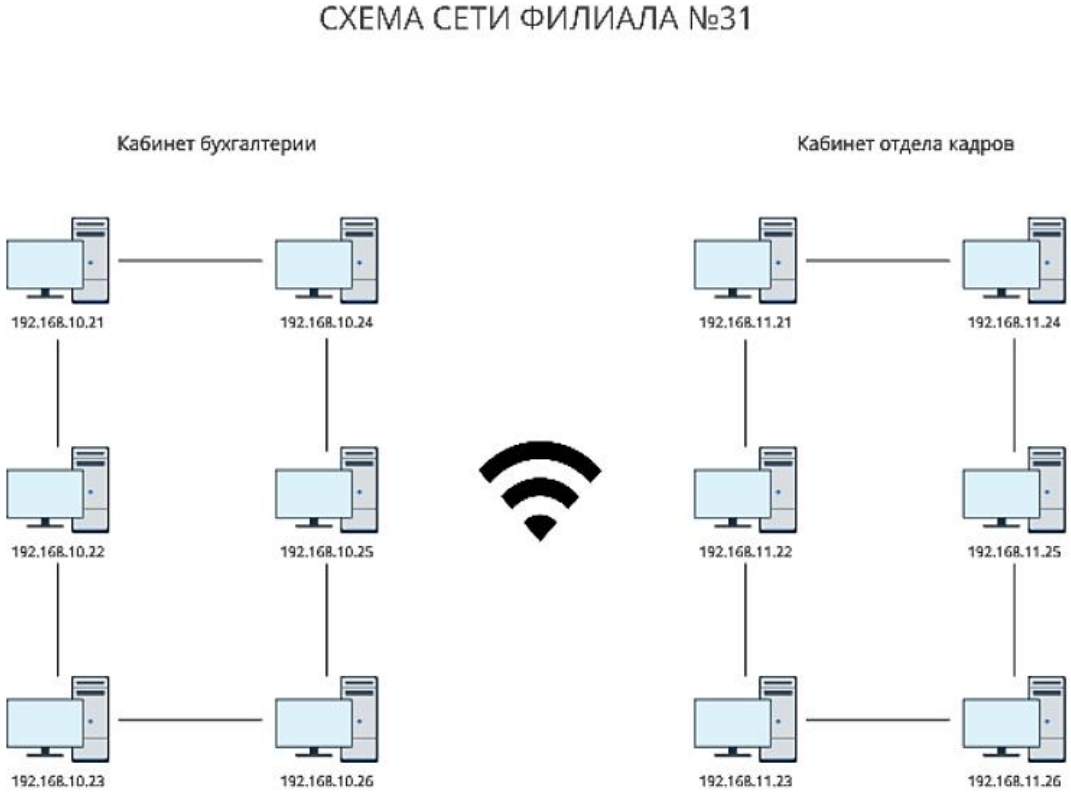

Рисунок 1 – Схема сети

К задаче прилагается:
файл «logs.txt»

Решение

Дана топология сети и файл logs.txt. На схеме мы ничего приметного не находим. Просто два отдела по 6 ПК с указанием IP-адресов.

Рассмотрим файл logs.txt. В нем можем увидеть урезанный дамп трафика с указанием времени, источника, назначения, протокола, длины и информации. Просмотрев весь трафик можно сделать вывод, что это ping-запросы, так как работает протокол ICMP и в информации передается информация, похожая на результаты выполнения команды ping.

При выполнении команды ping идет подсчет пакетов, отправленных от источника к получателю. Как, например, вот здесь:

```
└─# ping ya.ru
PING ya.ru (5.255.255.242) 56(84) bytes of data.
64 bytes from ya.ru (5.255.255.242): icmp_seq=1 ttl=240 time=12.5 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=2 ttl=240 time=6.16 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=3 ttl=240 time=6.65 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=4 ttl=240 time=6.64 ms
```

Однако в полученном нами файле, подсчет пакетов идет неправильно, начинаясь не с единицы, и, в принципе, имеет хаотичный порядок.

```
Время        Источник       Назначение     Протокол  Длина  Информация
10:44:01.022 192.168.10.25 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=213 ttl=128 time=15.9 ms
10:44:01.023 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=157 ttl=128 time=3.2 ms
10:44:01.024 192.168.10.25 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=25 ttl=128 time=24.5 ms
10:44:01.025 192.168.10.25 192.168.10.24 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.24: icmp_seq=15 ttl=128 time=7.0 ms
10:44:01.026 192.168.10.24 192.168.10.25 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.25: icmp_seq=208 ttl=128 time=5.2 ms
10:44:01.026 192.168.10.21 192.168.10.26 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.26: icmp_seq=98 ttl=64 time=27.8 ms
10:44:01.027 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=243 ttl=64 time=6.1 ms
10:44:01.029 192.168.10.24 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=5 ttl=128 time=2.0 ms
10:44:01.030 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=26 ttl=128 time=33.0 ms
10:44:01.030 192.168.10.24 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=18 ttl=128 time=34.1 ms
```

Значения в поле icmp_seq могут быть похожи на ASCII символы, которые скрыты в Decimal формате. Можем проверить эту гипотезу.

Согласно таблице ASCII есть непечатаемые символы от 0 до 31 и от 127 до 255. Исходя из этого все значения icmp_seq в данных диапазонах мы можем просто убрать из логирования.

Анализ log.txt: найдены ICMP-пакеты с icmp_seq

№	Источник	Назначение	Код	Символ
6	192.168.10.21	192.168.10.26	98	b
28	192.168.10.21	192.168.10.26	97	a
43	192.168.10.21	192.168.10.26	114	r
62	192.168.10.21	192.168.10.26	32	[пробел]
83	192.168.10.21	192.168.10.26	76	L
107	192.168.10.21	192.168.10.26	73	I
124	192.168.10.21	192.168.10.26	70	F
148	192.168.10.21	192.168.10.26	69	E

Ответ: bar LIFE (192.168.10.21 и 192.168.10.26)

Вариант 2

Отдел информационной безопасности компании «АС-Крипто» расследует деятельность внутреннего злоумышленника, который, по данным разведки, планирует встретиться со своим сообщником для передачи конфиденциальных данных. Команда аналитиков смогла перехватить часть трафика, где передавалось место встречи злоумышленников, однако понять, как именно оно передавалось им не удалось.

На основе имеющегося трафика, определите передаваемое место встречи и внутреннего злоумышленника.

В качестве ответа укажите место встречи и IP-адрес внутреннего злоумышленника.

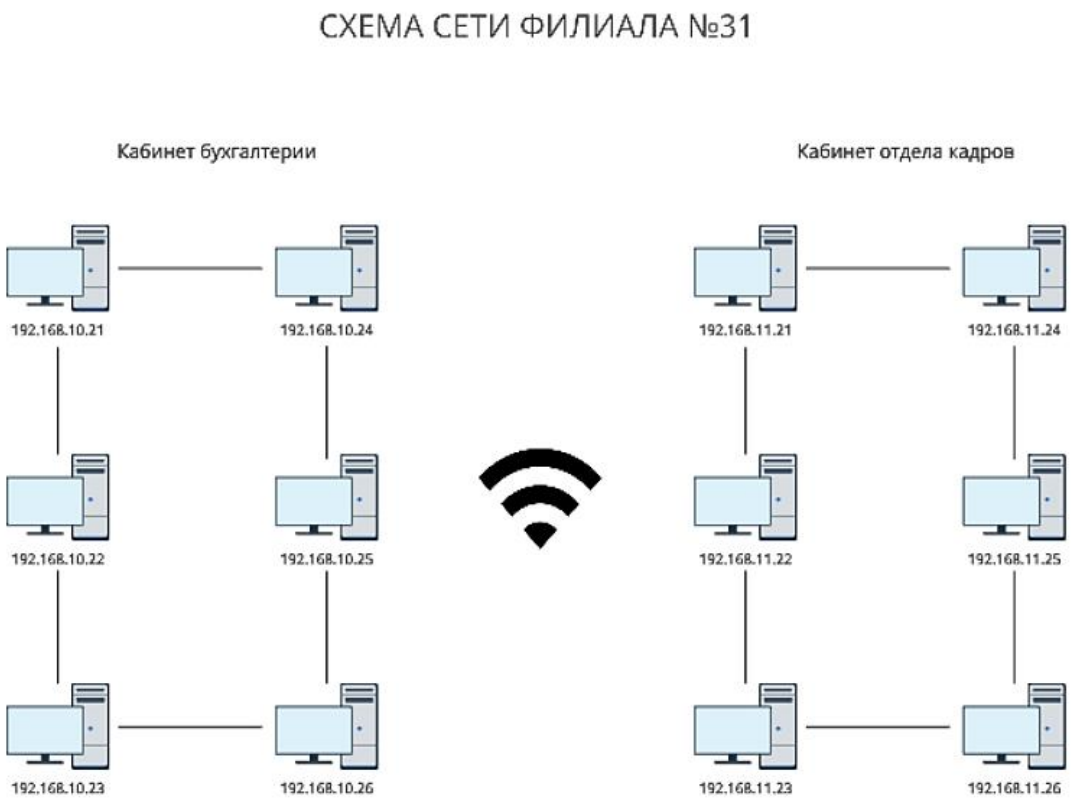

Рисунок 1 – Схема сети

К задаче прилагается:
файл «logs.txt»

Решение

Дана топология сети и файл logs.txt. На схеме мы ничего приметного не находим. Просто два отдела по 6 ПК с указанием IP-адресов.

Рассмотрим файл logs.txt. В нем можем увидеть урезанный дамп трафика с указанием времени, источника, назначения, протокола, длины и информации. Просмотрев весь трафик можно сделать вывод, что это ping-запросы, так как работает протокол ICMP и в информации передается информация, похожая на результаты выполнения команды ping.

При выполнении команды ping идет подсчет пакетов, отправленных от источника к получателю. Как, например, вот здесь:

```
└─# ping ya.ru
PING ya.ru (5.255.255.242) 56(84) bytes of data.
64 bytes from ya.ru (5.255.255.242): icmp_seq=1 ttl=240 time=12.5 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=2 ttl=240 time=6.16 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=3 ttl=240 time=6.65 ms
64 bytes from ya.ru (5.255.255.242): icmp_seq=4 ttl=240 time=6.64 ms
```

Однако в полученном нами файле, подсчет пакетов идет неправильно, начинаясь не с единицы, и, в принципе, имеет хаотичный порядок.

```
Время        Источник       Назначение     Протокол  Длина  Информация
10:44:01.022 192.168.10.25 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=213 ttl=128 time=15.9 ms
10:44:01.023 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=157 ttl=128 time=3.2 ms
10:44:01.024 192.168.10.25 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=25 ttl=128 time=24.5 ms
10:44:01.025 192.168.10.25 192.168.10.24 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.24: icmp_seq=15 ttl=128 time=7.0 ms
10:44:01.026 192.168.10.24 192.168.10.25 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.25: icmp_seq=208 ttl=128 time=5.2 ms
10:44:01.026 192.168.10.21 192.168.10.26 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.26: icmp_seq=98 ttl=64 time=27.8 ms
10:44:01.027 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=243 ttl=64 time=6.1 ms
10:44:01.029 192.168.10.24 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=5 ttl=128 time=2.0 ms
10:44:01.030 192.168.10.22 192.168.10.23 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.23: icmp_seq=26 ttl=128 time=33.0 ms
10:44:01.030 192.168.10.24 192.168.10.22 ICMP        64     Src port: 45000, Dst port: 45000, Data: 64 bytes from 192.168.10.22: icmp_seq=18 ttl=128 time=34.1 ms
```

Значения в поле icmp_seq могут быть похожи на ASCII символы, которые скрыты в Decimal формате. Можем проверить эту гипотезу.

Согласно таблице ASCII есть непечатаемые символы от 0 до 31 и от 127 до 255. Исходя из этого все значения icmp_seq в данных диапазонах мы можем просто убрать из логирования.

Анализ log.txt: найдены ICMP-пакеты с icmp_seq

№	Источник	Назначение	Код	Символ
10	192.168.10.22	192.168.10.25	97	a
26	192.168.10.22	192.168.10.25	114	r
40	192.168.10.22	192.168.10.25	101	e
60	192.168.10.22	192.168.10.25	110	n
76	192.168.10.22	192.168.10.25	97	a
103	192.168.10.22	192.168.10.25	32	[пробел]
123	192.168.10.22	192.168.10.25	70	F
140	192.168.10.22	192.168.10.25	49	1

Ответ: arena F1 (192.168.10.22 и 192.168.10.25)

Задача 5. Частотный анализ

Вариант 1

Отдел информационной безопасности компании «PostMail» перехватил подозрительное письмо, которое было отправлено с компьютера одного из сотрудников. После первичного анализа, не было установлено каким именно алгоритмом шифрования был зашифрован данный текст. Аналитики предоставили частоту появления букв в российских текстах (таблица 1) в надежде, что это поможет.

Определите, какое сообщение было передано и укажите в качестве ответа пароль администратора.

Таблица 1 – Частота встречаемости русских букв

Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота
А	6.72 %	**З**	1.12 %	**П**	2.52 %	**Ч**	1.68 %	**Я**	2.15 %
Б	1.59 %	**И**	7.84 %	**Р**	5.41 %	**Ш**	0.65 %		
В	4.38 %	**Й**	1.31 %	**С**	4.29 %	**Щ**	0.65 %		
Г	1.21 %	**К**	3.82 %	**Т**	7.74 %	**Ъ**	-		
Д	3.92 %	**Л**	3.73 %	**У**	3.17 %	**Ы**	1.21 %		
Е	7.93 %	**М**	2.43 %	**Ф**	0.65 %	**Ь**	2.52 %		
Ё	0.75 %	**Н**	5.88 %	**Х**	1.21 %	**Э**	0.47 %		
Ж	1.4 %	**О**	10,54 %	**Ц**	0.75 %	**Ю**	0.37 %		

Текст письма:

```
Зибъэкйкълц, ьилябсэ!
Бёмжиешобч л еэёч — ъйч, гшг ку б нжкэд. Яьл йъжб рэйкф йжкэё щшгйжъ. Гдшьб бн ъ кжк яэ кшвёбг аш щльгжв л ънжьш киб. Ёэ зжьъэьб — ч ёэ зижсл жщешёш!
Гшг б лйджъбдбйф: ъжк ьшёёуэ шьебёш. Шкшглв йкижыж зжйдэ зждлёжпб — ьж хкжыж зшкилдф нжьбк.
Джыбё: Ебижрёбпэёгж
Зшиждф: ЭъыэёбвГилкжвЙэкэъбг
Гйкшкб, зижъэиф, ёэк дб ичьже й йэиъэиёжв сэёгш — жнишёёбг бёжыьш зжьгшиедбъшэк эыж л ьъэиб. Мжёшибг ъжафеб зжкэеёээ, ш кж ъ гжибьжиэ йкжбк йкшиув мжкжхдэеэёк. Оэдув пшй зибьюкйч зичкшкфйч, эйдб йишщжкшэк! Сюдгёб кибяьу зж ъэёкбдчобжёёжв иэрюкгэ — хкж йбыёшд ьдч ежэыж пэджъэгш ъёлкиб. Хн, яшдф, пкж ёэдфач зижйкж зжаъжёбкф... Ля жпэёф йкижышч л ёбн йбйкэеш: ьшяэ щиэджг й пбзже киэщлэкйч, ьш эсю б збё-гжь ъъжьбкф пэиэа обмижълц гдшъбшклил. Щльф зиэьэдфёж жйкжижяэё — гшеэиу зжъйцьл, ьшяэ аш мшдфрбъжв гшикбёжв ъ нжддэ. Оуо! Б зжеёб: эйдб пкж — ъйю ёш кэщч йъшдчк. Ч пбйк, гшг йдэаш едшьэёош. Ш ку?..
Ашзжеёб: эйдб пкж — ч ёб зиб пюе. Л еэёч шдбщб яэдэаёжэ: ъ кл ёжпф щльл ъ «Слпфэе нъжйкэ» быишкф ъ ршнешку й Мюьжиже. Жё йъбьэкэдф! Гйкшкб, ёэ ашщльф ъугдцпбкф збкшёбэ ёш сбкгэ — кше ичьже йкжбк обиглдчигш, жёш ыльбк, гшг аъэиф. Хн, яшдф, пкж ёэдфач зижйкж ъщбкф гжь… Ёж л ёбн ьшяэ ёш гжмэъшигэ йкжбк щбжеэкибч!
Щльф кбрэ ъжьу, ёбяэ кишъу. Б ьш — зижъэиф, ёэк дб ъ гжибьжиэ ёжъжыж жнишёёбгш й щжижьжв. Ыжъжичк, жё щуърбв пэезбжё зж щжгйл. Жйкжижяёэв!
Яьл жкпюк.
```

К задаче прилагается:

файл «letter.txt»

Решение

Дан шифротекст и частотный анализ русского алфавита для расшифровки текста. Все что нам остается, это просто посчитать частоту букв в шифротексте.

Напишем программу, которая рассчитает нам частоту встречаемости букв в шифротексте

```
import re
from collections import Counter
```

```
import math

def frequency_analysis(text):
    # Извлекаем только русские буквы (а-я, ё), игнорируем регистр
    letters = re.findall(r'[а-яё]', text.lower())
    total = len(letters)

    if total == 0:
        print("В тексте не найдено ни одной русской буквы.")
        return

    freq = Counter(letters)
    sorted_items = freq.most_common()

    print(f"{'Буква':<6} {'Кол-во':<8} {'Процент':<10}")
    print("-" * 25)

    for letter, count in sorted_items:
        # Вычисляем процент
        percentage = (count / total) * 100
        # Округляем ВВЕРХ до сотых: ceil(percentage * 100) / 100
        percentage_ceil = math.ceil(percentage * 100) / 100
        # Форматируем до 2 знаков после запятой (всегда два знака: 3.1 → 3.10)
        formatted_percent = f"{percentage_ceil:.2f}"
        print(f"{letter:<6} {count:<8} {formatted_percent} %")

# Текст для анализа
TEXT = """Зибъэкйкълц, ьилябсэ!
Бёмжиешобч л еэёч — ъйч, гшг ку б нжкэд. Яьл йъжб рэйкф йжкэё щшгйжъ. Гдшьб бн ъ кжк яэ кшвёбг аш щльгжв л ънжьш киб. Ёэ зжьъэьб — ч ёэ зижсл жщешёш!
Гшг б лйджъбдбйф: ъжк ьшёёуэ шьебёш. Шкшглв йкижыж зжйдэ зждлёжпб — ьж хкжыж зшкилдф нжьбк.
Джыбё: Ебижрёбпэёгж
Зшиждф: ЭъыэёбвГилкжвЙэкэъбг
Гйкшкб, зижъэиф, ёэк дб ичьже й йэиъэиёжв сэёгш — жнишёёбг бёжыьш зжьгшиедбъшэк эыж л ьъэиб. Мжёшибг ъжафеб зжкэеёээ, ш кж ъ гжибьжиэ йкжбк йкшиув мжкжхдэеэёк. Оэдув пшй зибьюкйч зичкшкфйч, эйдб йишщжкшэк! Сюдгёб кибяьу зж ъэёкбдчобжёёжв иэрюкгэ — хкж йбыёшд ьдч ежэыж пэджъэгш ъёлкиб. Хн, яшдф, пкж ёэдфач зижйкж зжаъжёбкф... Ля жпэёф йкижышч л ёбн йбйкэеш: ьшяэ щиэджг й пбзже киэщлэкйч, ьш эсю б збё-гжь ъъжьбкф пэиэа обмижълц гдшъбшклил. Щльф зиэьэдфёж жйкжижяэё — гшеэиу зжъйцьл, ьшяэ аш мшдфрбъжв гшикбёжв ъ нжддэ. Оуо! Б зжеёб: эйдб пкж — ъйю ёш кэщч йъшдчк. Ч пбйк, гшг йдэаш едшьэёош. Ш ку?..
Ашзжеёб: эйдб пкж — ч ёб зиб пюе. Л еэёч шдбщб яэдэаёжэ: ъ кл ёжпф щльл ъ «Слпфэе нъжйкэ» быишкф ъ ршнешку й Мюьжиже. Жё йъбьэкэдф! Гйкшкб, ёэ ашщульф ъугдцпбкф збкшёбэ ёш сбкгэ — кше ичьже йкжбк обиглдчигш, жёш ыльбк, гшг аъэиф. Хн, яшдф, пкж ёэдфач зижйкж ъщбкф гжь… Ёж л ёбн ьшяэ ёш гжмэъшигэ йкжбк щбжеэкибч!
Щльф кбрэ ъжьу, ёбяэ кишъу. Б ьш — зижъэиф, ёэк дб ъ гжибьжиэ ёжъжыж жнишёёбгш й щжижьжв. Ыжъжичк, жё щуърбв пэезбжё зж щжгйл. Жйкжижяёэв!
Яьл жкпюк."""

frequency_analysis(TEXT)
```

Результат работы программы:

```
Буква  Кол-во   Процент
-------------------------
ж      113      10.54 %
э      85       7.93 %
б      84       7.83 %
к      83       7.74 %
ш      72       6.72 %
ё      63       5.88 %
```

```
и       58      5.41 %
ъ       47      4.39 %
й       46      4.29 %
ь       42      3.92 %
г       41      3.83 %
д       40      3.73 %
л       34      3.17 %
з       27      2.52 %
ф       27      2.52 %
е       26      2.43 %
ч       23      2.15 %
п       18      1.68 %
щ       17      1.59 %
я       15      1.40 %
у       14      1.31 %
в       14      1.22 %
н       13      1.22 %
ы       13      1.22 %
а       12      1.12 %
о       8       0.75 %
ю       8       0.75 %
с       7       0.66 %
м       7       0.66 %
р       7       0.66 %
х       5       0.47 %
ц       4       0.38 %
```

Видно, что частота появления букв совпадает с той, что нам дано в задании, поэтому необходимо просто заменить все буквы, согласно частоте. Однако, у нас есть буквы, которые появляются одинаково частно. В этом случае придется подбирать вручную, когда перенесем уже известные буквы.

Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста
А	Ш	З	А	П	З/Ф	Ч	П	Я	Ч
Б	Щ	И	Б	Р	И	Ш	С/М/Р		
В	Ъ	Й	У	С	Й	Щ	С/М/Р		
Г	В/Н/Ы	К	Г	Т	К	Ъ	-		
Д	Ь	Л	Д	У	Л	Ы	В/Н/Ы		
Е	Э	М	Е	Ф	С/М/Р	Ь	З/Ф		
Ё	О/Ю	Н	Ё	Х	В/Н/Ы	Э	Х		
Ж	Я	О	Ж	Ц	О/Ю	Ю	Ц		

Заменим текст, используя только известные буквы, неизвестные оставим под знаком – «?».

?риветствую, дружи?е!

Ин?орма?ия у меня — вся, как тй и ?отел. Жду свои ?ест? сотен баксов. Клади и? в тот же та?ник за будко? у в?ода три. Не ?одведи — я не ?ро?у обмана!

Как и условилис?: вот даннйе админа. Атаку? стро?о ?осле ?олуночи — до это?о ?атрул? ?одит.

Ло?ин: Миро?ниченко

?арол?: Ев?ени?Круто?Сетевик

Кстати, ?ровер?, нет ли рядом с серверно? ?енка — о?ранник ино?да ?одкармливает е?о у двери. ?онарик воз?ми ?отемнее, а то в коридоре стоит старй? ?отоэлемент. ?елй? час ?рид?тся ?рятат?ся, если сработает! ??лкни триждй ?о вентиля?ионно? ре??тке — это си?нал для мое?о человека внутри. Э?, жал?, что нел?зя ?росто ?озвонит?... Уж очен? стро?ая у ни? система: даже брелок с чи?ом требуется, да е?? и ?ин-код вводит? через ?и?ровую клавиатуру. Буд? ?редел?но осторожен — камерй ?овсюду, даже за ?ал??иво? картино? в ?олле. ?й?! И ?омни: если что — вс? на тебя свалят. Я чист, как слеза младен?а. А тй?..

За?омни: если что — я ни ?ри ч?м. У меня алиби железное: в ту ноч? буду в «?уч?ем ?восте» и?рат? в ?а?матй с ??дором. Он свидетел?! Кстати, не забйуд?

вйключит? ?итание на ?итке — там рядом стоит ?иркулярка, она ?удит, как звер?. Э?, жал?, что нел?зя ?росто вбит? код… Но у ни? даже на ко?еварке стоит биометрия!

Буд? ти?е водй, ниже травй. И да — ?ровер?, нет ли в коридоре ново?о о?ранника с бородо?. ?оворят, он бйв?и? чем?ион ?о боксу. Осторожне?!

Жду отч?т.

Подставив буквы по смыслу, получим:

Приветствую, дружище!

Информация у меня — вся, как ты и хотел. Жду свои шесть сотен баксов. Клади их в тот же тайник за будкой у входа три. Не подведи — я не прощу обмана!

Как и условились: вот данные админа. Атакуй строго после полуночи — до этого патруль ходит.

Логин: Мирошниченко

Пароль: **ЕвгенийКрутойСетевик**

Кстати, проверь, нет ли рядом с серверной щенка — охранник иногда подкармливает его у двери. Фонарик возьми потемнее, а то в коридоре стоит старый фотоэлемент. Целый час придётся прятаться, если сработает! Щёлкни трижды по вентиляционной решётке — это сигнал для моего человека внутри. Эх, жаль, что нельзя просто позвонить... Уж очень строгая у них система: даже брелок с чипом требуется, да ещё и пин-код вводить через цифровую клавиатуру. Будь предельно осторожен — камеры повсюду, даже за фальшивой картиной в холле. Цыц! И помни: если что — всё на тебя свалят. Я чист, как слеза младенца. А ты?..

Запомни: если что — я ни при чём. У меня алиби железное: в ту ночь буду в «Щучьем хвосте» играть в шахматы с Фёдором. Он свидетель! Кстати, не забудь выключить питание на щитке — там рядом стоит циркулярка, она гудит, как зверь. Эх, жаль, что нельзя просто вбить код… Но у них даже на кофеварке стоит биометрия!

Будь тише воды, ниже травы. И да — проверь, нет ли в коридоре нового охранника с бородой. Говорят, он бывший чемпион по боксу. Осторожней!

Жду отчёт.

Ответ: ЕвгенийКрутойСетевик

Вариант 2

Отдел информационной безопасности компании «PostMail» перехватил подозрительное письмо, которое было отправлено с компьютера одного из сотрудников. После первичного анализа, не было установлено каким именно алгоритмом шифрования был зашифрован данный текст. Поэтому, Вам предоставлена частота появления букв в российских текстах (Таблица 1), для расшифровки текста. В качестве ответа необходимо предоставить пароль администратора.

Таблица 1 – Частота встречаемости русских букв

Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота
А	6,81 %	**З**	1.12 %	**П**	2.52 %	**Ч**	1.68 %	**Я**	2.15 %
Б	1.59 %	**И**	7,93 %	**Р**	5.5 %	**Ш**	0.65 %		
В	4.29 %	**Й**	1.31 %	**С**	4.2 %	**Щ**	0.65 %		
Г	1.12 %	**К**	3.73 %	**Т**	7,74 %	**Ъ**	-		
Д	3.92 %	**Л**	3.82 %	**У**	3.17 %	**Ы**	1.21 %		
Е	7,74 %	**М**	2.43 %	**Ф**	0.65 %	**Ь**	2.52 %		
Ё	0.75 %	**Н**	5.97 %	**Х**	1.21 %	**Э**	0.47 %		
Ж	1.49 %	**О**	10,54 %	**Ц**	0.75 %	**Ю**	0.37 %		

Текст письма:

Абщтхгвгтдо, фбдчщйх!

Щюеябэржщп д эхюп — твп, ыры гл щ ёягхь. Чфд втящ ихвгм вягхю срывят. Ыьрфщ щё т гяг чх гръющы шр сдфыяъ д тёяфр гбщ. Юх аяфтхфщ — п юх абяйд ясэрюр!

Ыры щ двьятщьщвм: тяг фрююлх рфэщюр. Ргрыдъ вгбяуя аявьх аяьдюязщ — фя нгяуя аргбдьм ёяфщг.

```
Ьяущю: Эщбяиющзхюыя
Арбяьм: ТщгрьщъЫбдгяъЩючхюхб
```

Ывгргщ, абятхбм, юхг ьщ бпфяэ в вхбтхбюяъ йхюыр — яёбрююшы щояуфр аяфырбэьштрхг хуя д фтхбщ. Еяюрбшы тяшмэщ аягхэюхх, р гя т ыябщфябх вгящг вгрблъ еягяньхэхюг. Жхьлъ зрв абщфцгвп абпгргмвп, хвьщ вбрсягрхг! Йцьыющ гбщчфл ая тхюгщьпжщяююяъ бхицгых — нгя вщуюрь фьп эяхуя зхьятхыр тюдгбщ. Нё, чрьм, згя юхьмшп абявгя аяштяющгм... Дч язхюм вгбяурп д ющё вщвгхэр: фрчх сбхьяы в зщаяэ гбхсдхгвп, фр хйц щ ащю-ыяф ттяфщгм зхбхш жщебятдо ыьртщргдбд. Сдфм абхфхьмюя явгябячхю — ырэхбл аятвофд, фрчх шр ерьмищтяъ ырбгщюяъ т ёяььх. Жлж! Щ аяэющ: хвьщ згя — твц юр гхсп втрьпг. П зщвг, ыры вьхшр эьрфхюжр. Р гл?..

Шраяэющ: хвьщ згя — п ющ абщ зцэ. Д эхюп рьщсщ чхьхшюях: т гд юязм сдфд т «Йдзмхэ ётявгх» щубргм т ирёэргл в Ецфябяэ. Яю втщфхгхьм! Ывгргщ, юх шрсдфм тлыьозщгм ащгрющх юр йщгых — грэ бпфяэ вгящг жщбыдьпбыр, яюр удфщг, ыры штхбм. Нё, чрьм, згя юхьмшп абявгя тсщгм ыяф… Юя д ющё фрчх юр ыяехтрбых вгящг сщяэхгбщп!

Сдфм гщих тяфл, ющчх гбртл. Щ фр — абятхбм, юхг ьщ т ыябщфябх юятяуя яёбрююшыр в сябяфяъ. Уятябпг, яю слтищъ зхэащяю ая сяывд. Явгябячюхъ!

Чфд ягзцг.

К задаче прилагается:
файл «letter.txt»

Решение

Дан шифротекст и частотный анализ русского алфавита для расшифровки текста. Все что нам остается, это просто посчитать частоту букв в шифротексте.

Напишем программу, которая рассчитает нам частоту встречаемости букв в шифротексте

```
import re
from collections import Counter
import math

def frequency_analysis(text):
    # Извлекаем только русские буквы (а-я, ё), игнорируем регистр
    letters = re.findall(r'[а-яё]', text.lower())
    total = len(letters)

    if total == 0:
        print("В тексте не найдено ни одной русской буквы.")
        return

    freq = Counter(letters)
    sorted_items = freq.most_common()

    print(f"{'Буква':<6} {'Кол-во':<8} {'Процент':<10}")
    print("-" * 25)

    for letter, count in sorted_items:
        # Вычисляем процент
        percentage = (count / total) * 100
        # Округляем ВВЕРХ до сотых: ceil(percentage * 100) / 100
        percentage_ceil = math.ceil(percentage * 100) / 100
        # Форматируем до 2 знаков после запятой (всегда два знака: 3.1 → 3.10)
        formatted_percent = f"{percentage_ceil:.2f}"
        print(f"{letter:<6} {count:<8} {formatted_percent} %")

# Текст для анализа
TEXT = """ Абщтхгвгтдо, фбдчщйх!
Щюеябэржщп д эхюп — твп, ыры гл щ ёягхь. Чфд втящ ихвгм вягхю срывят. Ыьрфщ щё т гяг чх гръющы шр сдфыяъ д тёяфр гбщ. Юх аяфтхфщ — п юх абяйд ясэрюр!
Ыры щ двьятщьщвм: тяг фрююлх рфэщюр. Ргрыдъ вгбяуя аявьх аяьдюязщ — фя нгяуя аргбдьм ёяфщг.
Ьяущю: Эщбяиющзхюыя
```

```
Арбяьм: ТщгрьщъЫбдгяъЩючхюхб
Ывгргщ, абятхбм, юхг ьщ бпфяэ в вхбтхбюяъ йхюыр — яёбрююцы щюяуфр аяфырбэьщтрхг хуя д фтхбщ. Еяюрбцы тяшмэщ аягхэюхх, р гя т ыябщфябх вгящг вгрблъ еягяньхэхюг. Жхьлъ зрв абщфцгвп абпгргмвп, хвьщ вбрсягрхг! Йцьыющ гбщчфл ая тхюгщьпжщяююяъ бхицгых — нгя вщуюрь фьп эяхуя зхьятхыр тюдгбщ. Нё, чрьм, згя юхьмшп абявгя аяштяющгм... Дч язхюм вгбяурп д ющё вщвгхэр: фрчх сбхьяы в зщаяэ гбхсдхгвп, фр хйц щ ащю-ыяф ттяфщгм зхбхш жщебятдо ыьртщргдбд. Сдфм абхфхьмюя явгябячхю — ырэхбл аятвофд, фрчх шр ерьмищтяъ ырбгщюяъ т ёяььх. Жлж! Щ аяэющ: хвьщ згя — твц юр гхсп втрьпг. П зщвг, ыры вьхшр эьрфхюжр. Р гл?..
Шраяэющ: хвьщ згя — п ющ абщ зцэ. Д эхюп рьщсщ чхьхшюях: т гд юязм сдфд т «Йдзмхэ ётявгх» щубргм т ирёэргл в Ецфябяэ. Яю втщфхгхьм! Ывгргщ, юх шрсдфм тлыьозщгм ащгрющх юр йщгых — грэ бпфяэ вгящг жщбыдьпбыр, яюр удфщг, ыры штхбм. Нё, чрьм, згя юхьмшп абявгя тсщгм ыяф… Юя д ющё фрчх юр ыяехтрбых вгящг сщяэхгбщп!
Сдфм гщих тяфл, ющчх гбртл. Щ фр — абятхбм, юхг ьщ т ыябщфябх юятяуя яёбрююцыр в сябяфяъ. Уятябпг, яю слтищъ зхэащяю ая сяывд. Явгябячюхъ!
Чфд ягзцг."""

frequency_analysis(TEXT)
```

Результат работы программы:

```
Буква  Кол-во   Процент
-------------------------
я      113      10.55 %
щ      85       7.93 %
х      83       7.75 %
г      83       7.75 %
р      73       6.81 %
ю      64       5.98 %
б      59       5.51 %
т      46       4.30 %
в      45       4.20 %
ф      42       3.92 %
ь      41       3.83 %
ы      40       3.74 %
д      34       3.18 %
а      27       2.52 %
м      27       2.52 %
э      26       2.43 %
п      23       2.15 %
з      18       1.68 %
с      17       1.59 %
ч      16       1.50 %
ъ      14       1.31 %
л      13       1.22 %
ё      13       1.22 %
ш      12       1.12 %
у      12       1.12 %
ж      8        0.75 %
ц      8        0.75 %
й      7        0.66 %
е      7        0.66 %
и      7        0.66 %
н      5        0.47 %
о      4        0.38 %
```

Видно, что частота появления букв совпадает с той, что нам дано в задании, поэтому необходимо просто заменить все буквы, согласно частоте. Однако, у нас есть буквы, которые появляются одинаково частно. В этом случае придется подбирать вручную, когда перенесем уже известные буквы.

Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста	Буква открытого текста	Буква шифротекста
А	Р	З	Ш/У	П	А/М	Ч	З	Я	П
Б	С	И	Щ	Р	Б	Ш	Й/Е/И		
В	Т	Й	Ъ	С	В	Щ	Й/Е/И		
Г	Ш/У	К	Ы	Т	Х/Г	Ь	А/М		
Д	Ф	Л	Ь	У	Д	Ы	Л/Ё		
Е	Х/Г	М	Э	Ф	Й/Е/И	Ъ	-		
Ё	Ж/Ц	Н	Ю	Х	Л/Ё	Э	Н		
Ж	Ч	О	Я	Ц	Ж/Ц	Ю	О		

Заменим текст, используя только известные буквы, неизвестные оставим под знаком – «?».

Прив??с?вую, дружи??!

Ин?орма?ия у м?ня — вся, как ?? и ?о??л. Жду свои ??с?? со??н баксов. Клади и? в ?о? ж? ?айник ?а будкой у в?ода ?ри. Н? ?одв?ди — я н? ?ро?у обмана!

Как и условились?: во? данн?? админа. А?акуй с?ро?о ?осл? ?олуночи — до э?о?о ?а?рул? ?оди?.

Ло?ин: Миро?нич?нко

Парол?: Ви?алийКру?ойИнж?н?р

Кс?а?и, ?ров?р?, н?? ли рядом с с?рв?рной ??нка — о?ранник ино?да ?одкармлива?? ??о у дв?ри. Фонарик во??ми ?о??мн??, а ?о в коридор? с?ои? с?ар?й ?о?оэл?м?н?. Ц?л?й час ?рид??ся ?ря?а??ся, ?сли срабо?а??! Щ?лкни ?рижд? ?о в?н?иля?ионной р????к? — э?о си?нал для мо??о ч?лов?ка вну?ри. Э?, жал?, ч?о н?л??я ?рос?о ?о?вони??... Уж оч?н? с?ро?ая у ни? сис??ма: даж? бр?лок с чи?ом ?р?бу??ся, да ??? и ?ин-код вводи?? ч?р?? ?и?ровую клавиа?уру. Буд? ?р?д?л?но ос?орож?н — кам?р? ?овсюду, даж? ?а ?ал??ивой кар?иной в ?олл?. Ц??! И ?омни: ?сли ч?о — вс? на ??бя сваля?. Я чис?, как сл??а млад?н?а. А ???..

За?омни: ?сли ч?о — я ни ?ри ч?м. У м?ня алиби ж?л??но?: в ?у ноч? буду в «Щуч??м ?вос??» и?ра?? в ?а?ма?? с Ф?дором. Он свид???л?! Кс?а?и, н? ?абуд? в?ключи?? ?и?ани? на ?и?к? — ?ам рядом с?ои? ?иркулярка, она ?уди?, как ?в?р?. Э?, жал?, ч?о н?л??я ?рос?о вби?? код… Но у ни? даж? на ко??варк? с?ои? биом??рия!

Буд? ?и?? вод?, ниж? ?рав?. И да — ?ров?р?, н?? ли в коридор? ново?о о?ранника с бородой. Говоря?, он б?в?ий ч?м?ион ?о боксу. Ос?орожн?й!

Жду о?ч??.

Подставив буквы по смыслу, получим:

Приветствую, дружище!

Информация у меня — вся, как ты и хотел. Жду свои шесть сотен баксов. Клади их в тот же тайник за будкой у входа три. Не подведи — я не прощу обмана!

Как и условились: вот данные админа. Атакуй строго после полуночи — до этого патруль ходит.

Логин: Мирошниченко

Пароль: **ВиталийКрутойИнженер**

Кстати, проверь, нет ли рядом с серверной щенка — охранник иногда подкармливает его у двери. Фонарик возьми потемнее, а то в коридоре стоит старый фотоэлемент. Целый час придётся прятаться, если сработает! Щёлкни трижды по вентиляционной решётке — это сигнал для моего человека внутри. Эх, жаль, что нельзя просто позвонить... Уж очень строгая у них система: даже брелок с чипом требуется, да ещё и пин-код вводить через цифровую клавиатуру. Будь предельно осторожен — камеры повсюду, даже за фальшивой картиной в холле. Цыц! И помни: если что — всё на тебя свалят. Я чист, как слеза младенца. А ты?..

Запомни: если что — я ни при чём. У меня алиби железное: в ту ночь буду в «Щучьем хвосте» играть в шахматы с Фёдором. Он свидетель! Кстати, не забудь выключить питание на щитке — там рядом стоит циркулярка, она гудит, как зверь. Эх, жаль, что нельзя просто вбить код… Но у них даже на кофеварке стоит биометрия!

Будь тише воды, ниже травы. И да — проверь, нет ли в коридоре нового охранника с бородой. Говорят, он бывший чемпион по боксу. Осторожней!

Жду отчёт.

Ответ: ВиталийКрутойИнженер