САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №6

по курсу «Алгоритмы и структуры данных»

Тема: Хеширование. Хеш-таблицы

Выполнила:

Просветова Валерия Дмитриевна

К3141

Проверил:

Афанасьев.А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

Задача №1: Множество

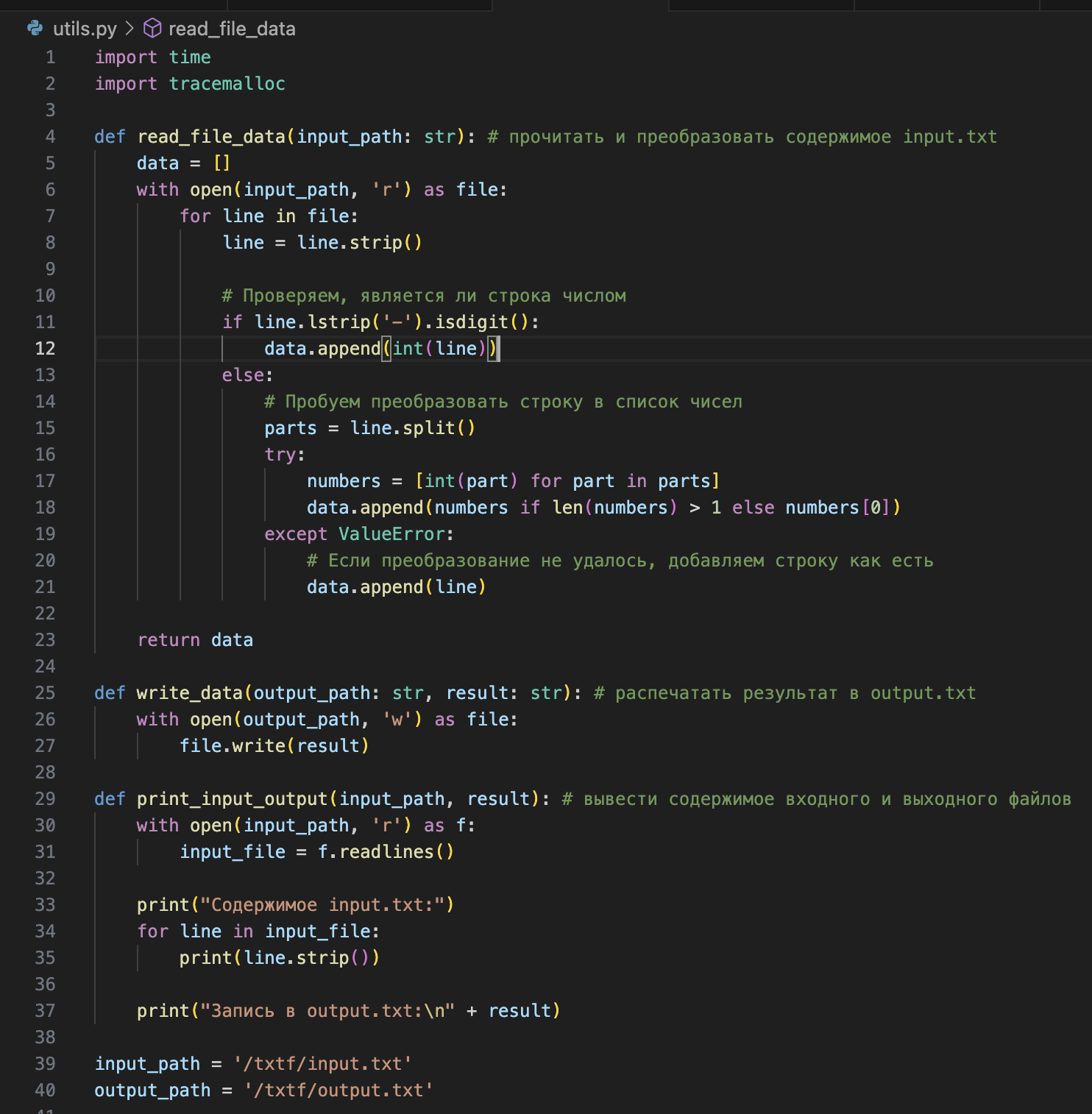
Задача №4: Прошитый ассоциативный массив

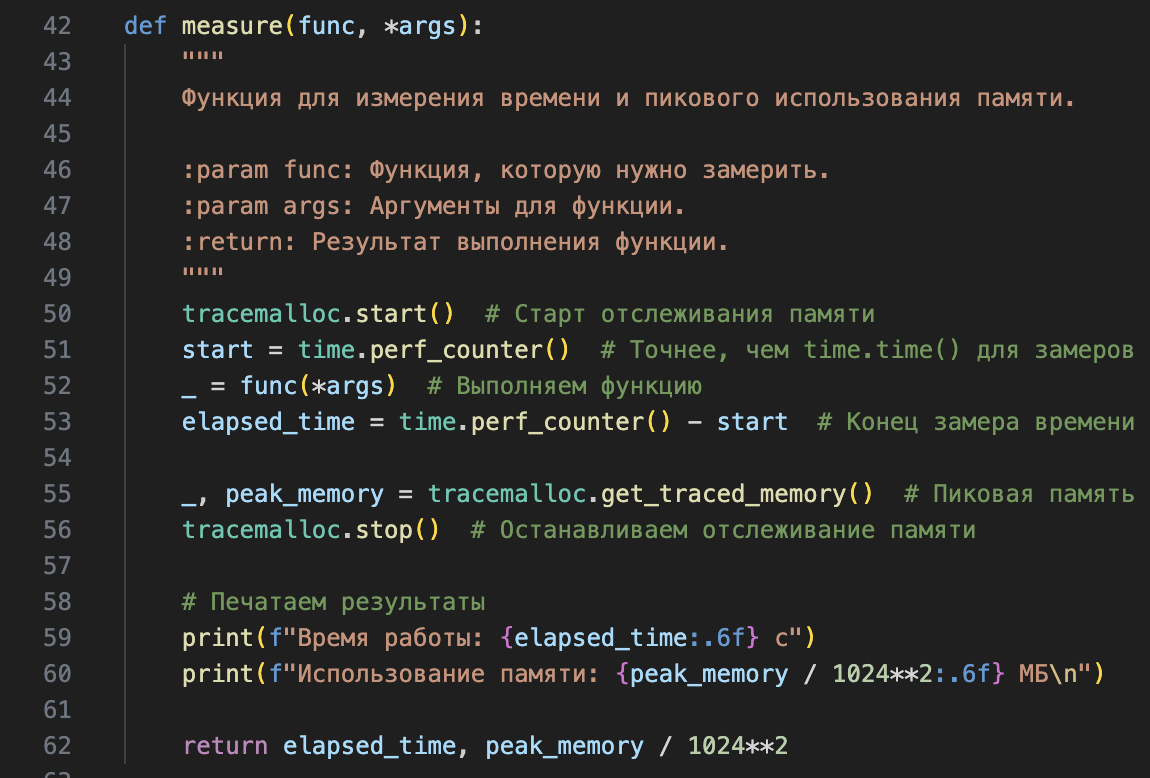
Задача №5: Выборы в США

Задача №7: Фибоначчи возвращается

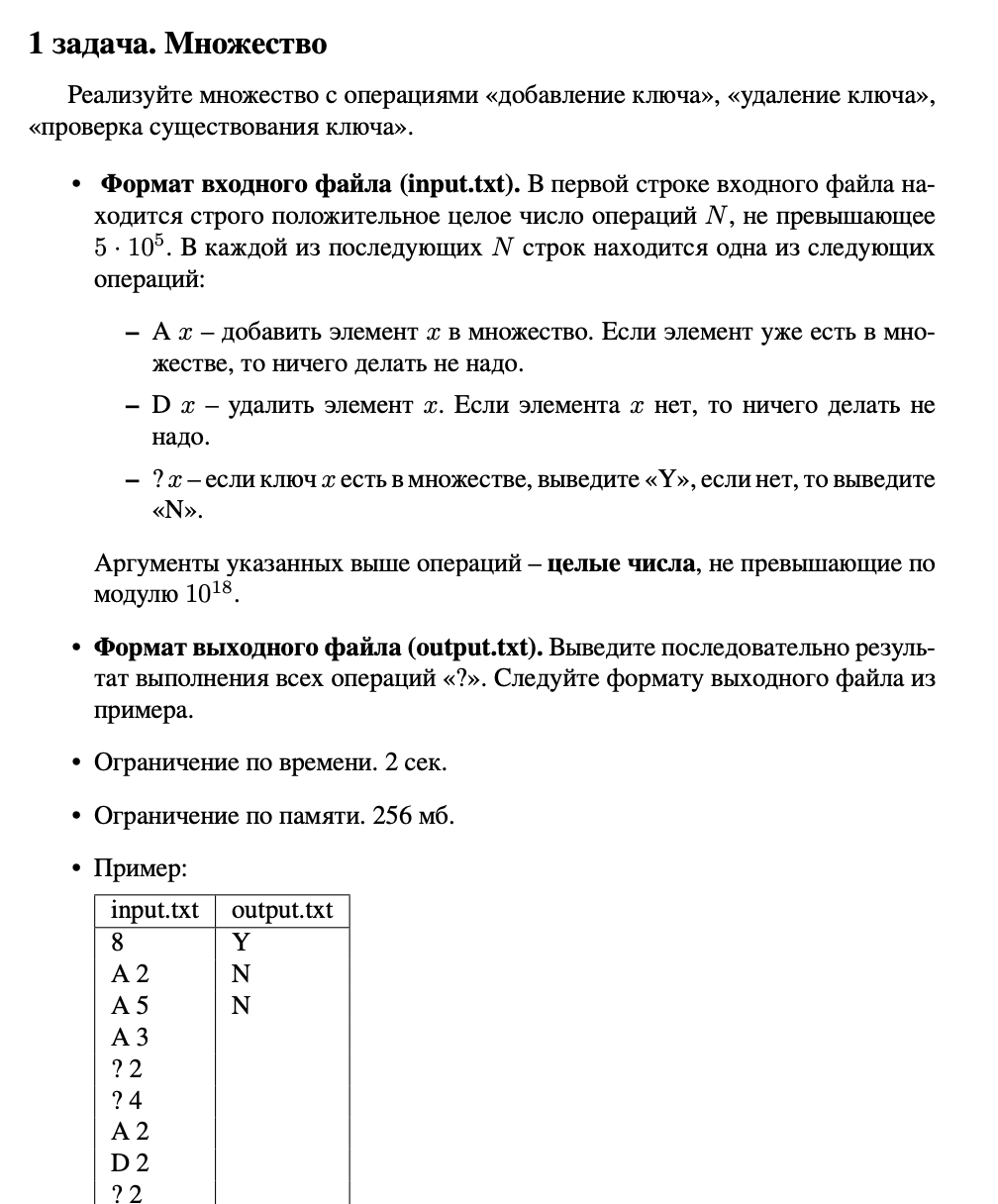
Вывод

Использовала файл utils.py для исключения повторения строчек кода:

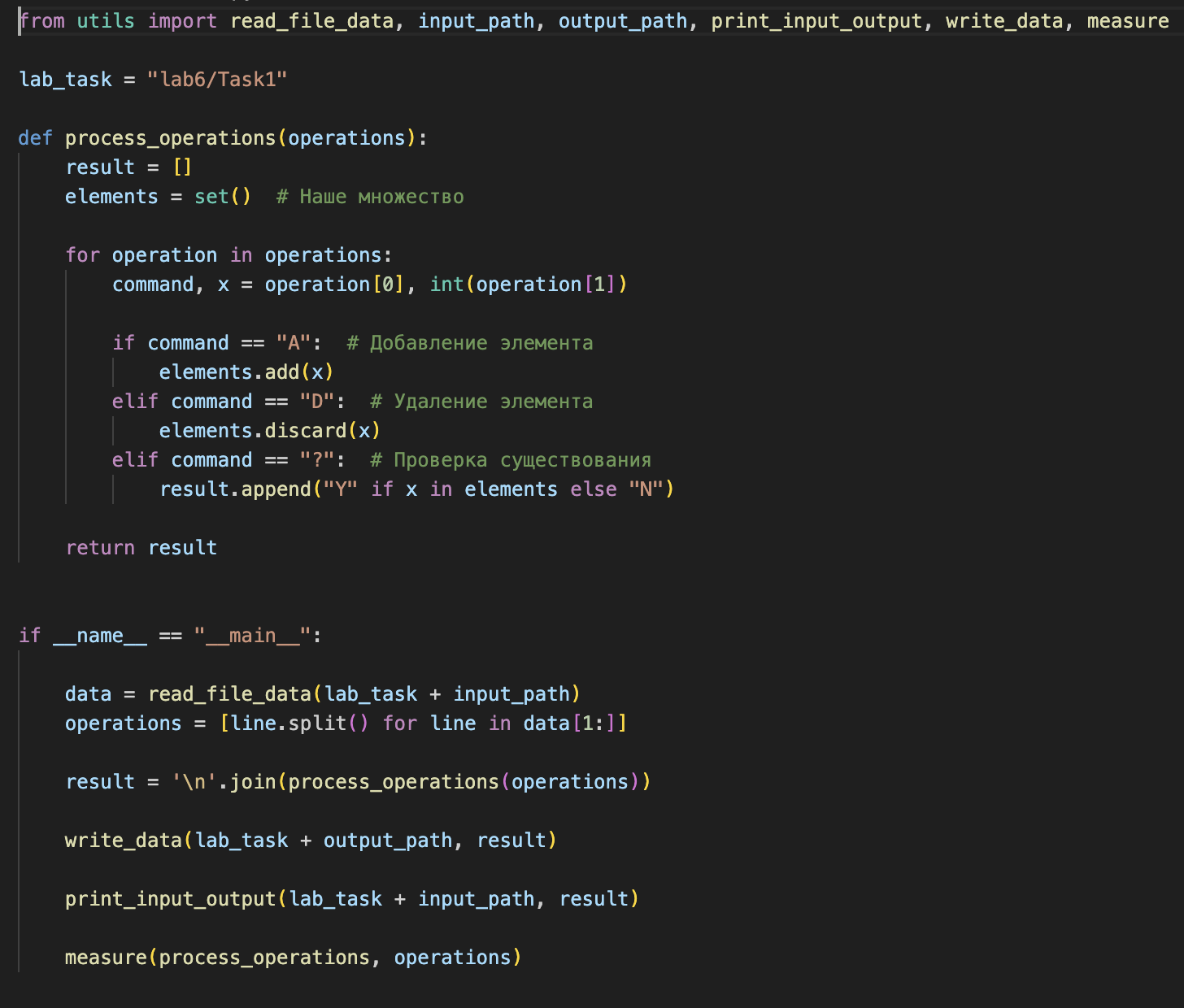




Задача №1:



Решение:



**Шаг 1: Определение функции**

def process\_operations(operations):

Эта строка определяет функцию process\_operations, которая принимает один аргумент operations.

**Шаг 2: Инициализация переменных**

result = [] elements = set() # Наше множество

Эти две строки инициализируют две переменные:

* result - пустой список, который будет использоваться для хранения результатов обработки операций.
* elements - пустое множество, которое будет использоваться для хранения элементов.

**Шаг 3: Цикл по операциям**

for operation in operations:

Эта строка начинает цикл по операциям, которые переданы в функцию process\_operations. Каждая операция представляет собой кортеж из двух элементов: команды и значения.

**Шаг 4: Разбор операции**

command, x = operation[0], int(operation[1])

Эта строка разбирает операцию на две части:

* command - это первая часть операции, которая представляет собой команду (например, "A", "D" или "?").
* x - это вторая часть операции, которая представляет собой значение (например, целое число).

**Шаг 5: Обработка команды**

if command == "A": # Добавление элемента elements.add(x)

Если команда равна "A", то элемент x добавляется в множество elements.

elif command == "D": # Удаление элемента elements.discard(x)

Если команда равна "D", то элемент x удаляется из множества elements.

elif command == "?": # Проверка существования result.append("Y" if x in elements else "N")

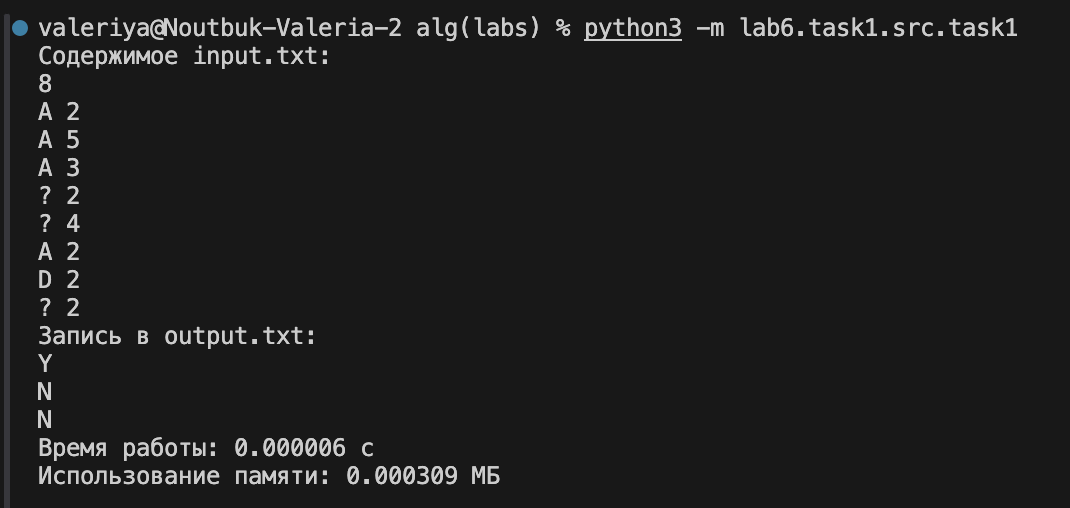
Если команда равна "?", то проверяется, существует ли элемент x в множестве elements. Если элемент существует, то в список result добавляется строка "Y", иначе добавляется строка "N".

**Шаг 6: Возвращение результата**

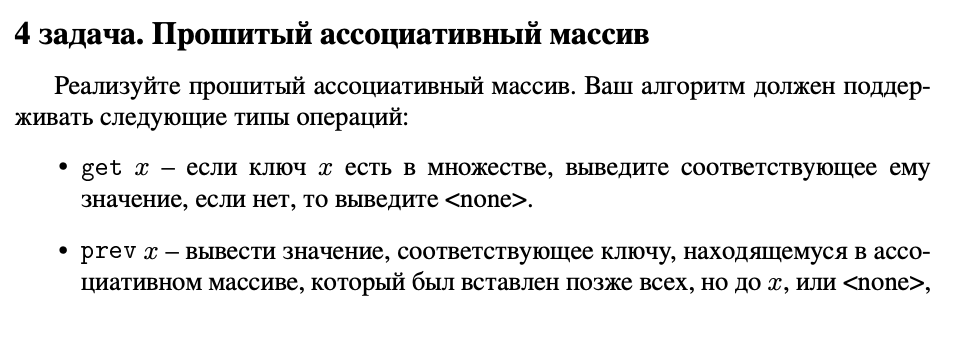
return result

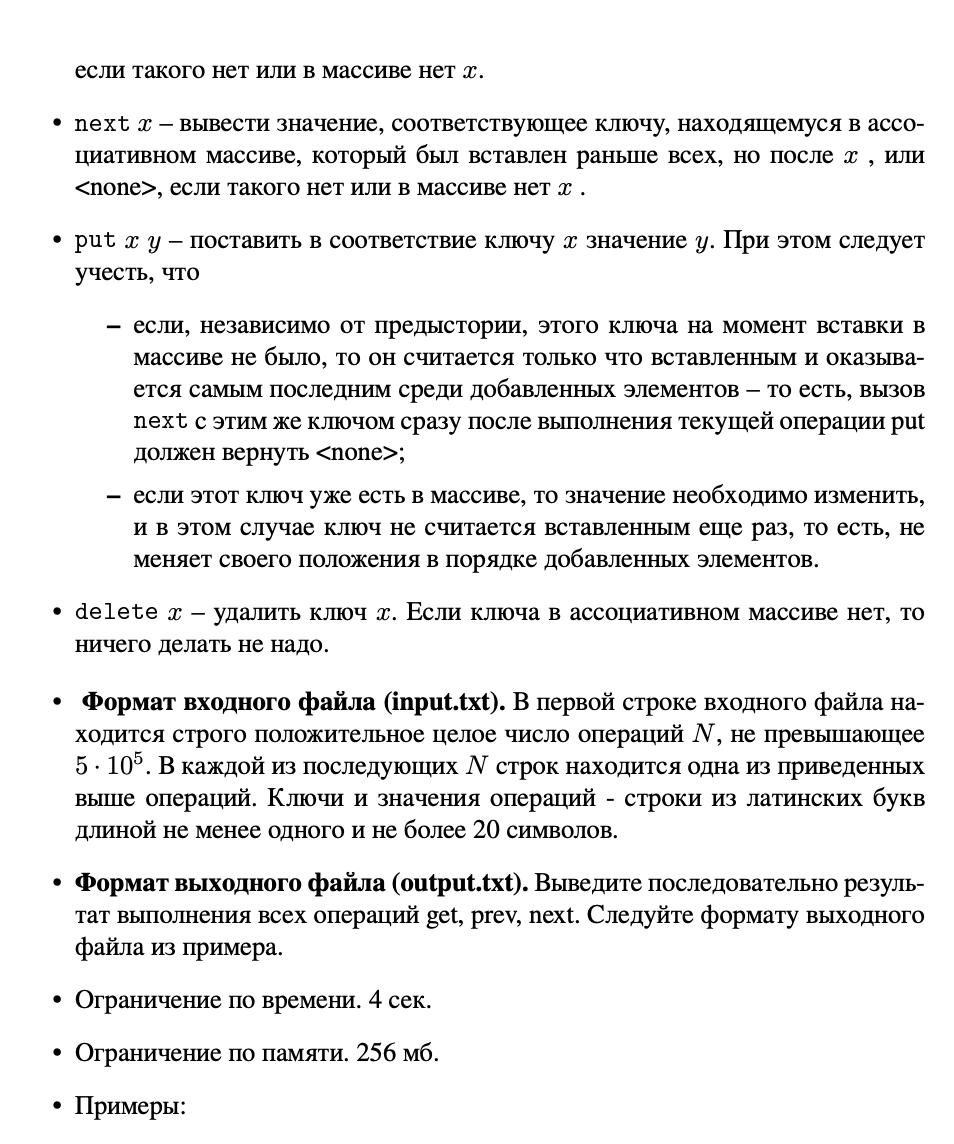
После обработки всех операций функция возвращает список result, который содержит результаты проверки существования элементов.

Запустила код и получила результат:

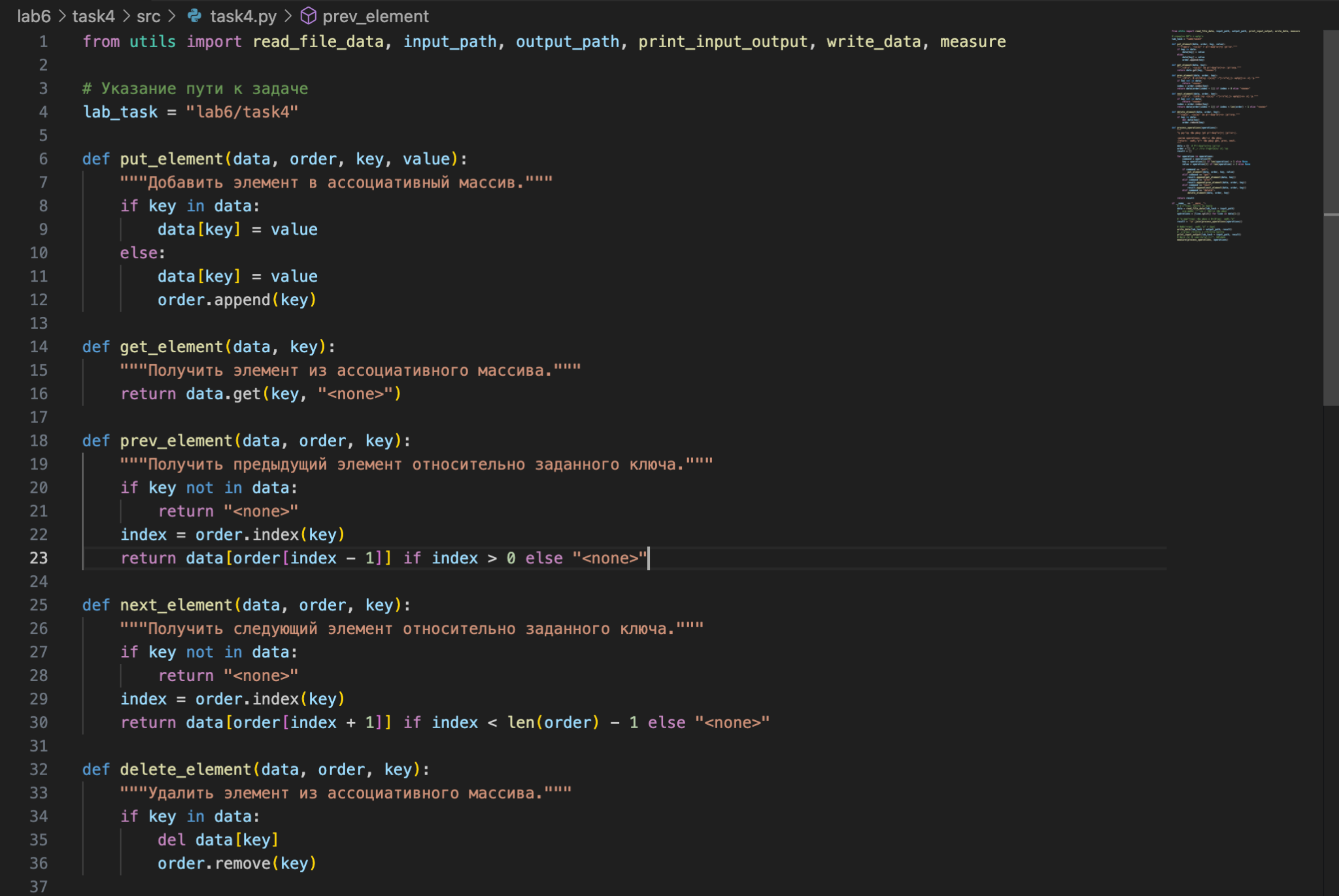


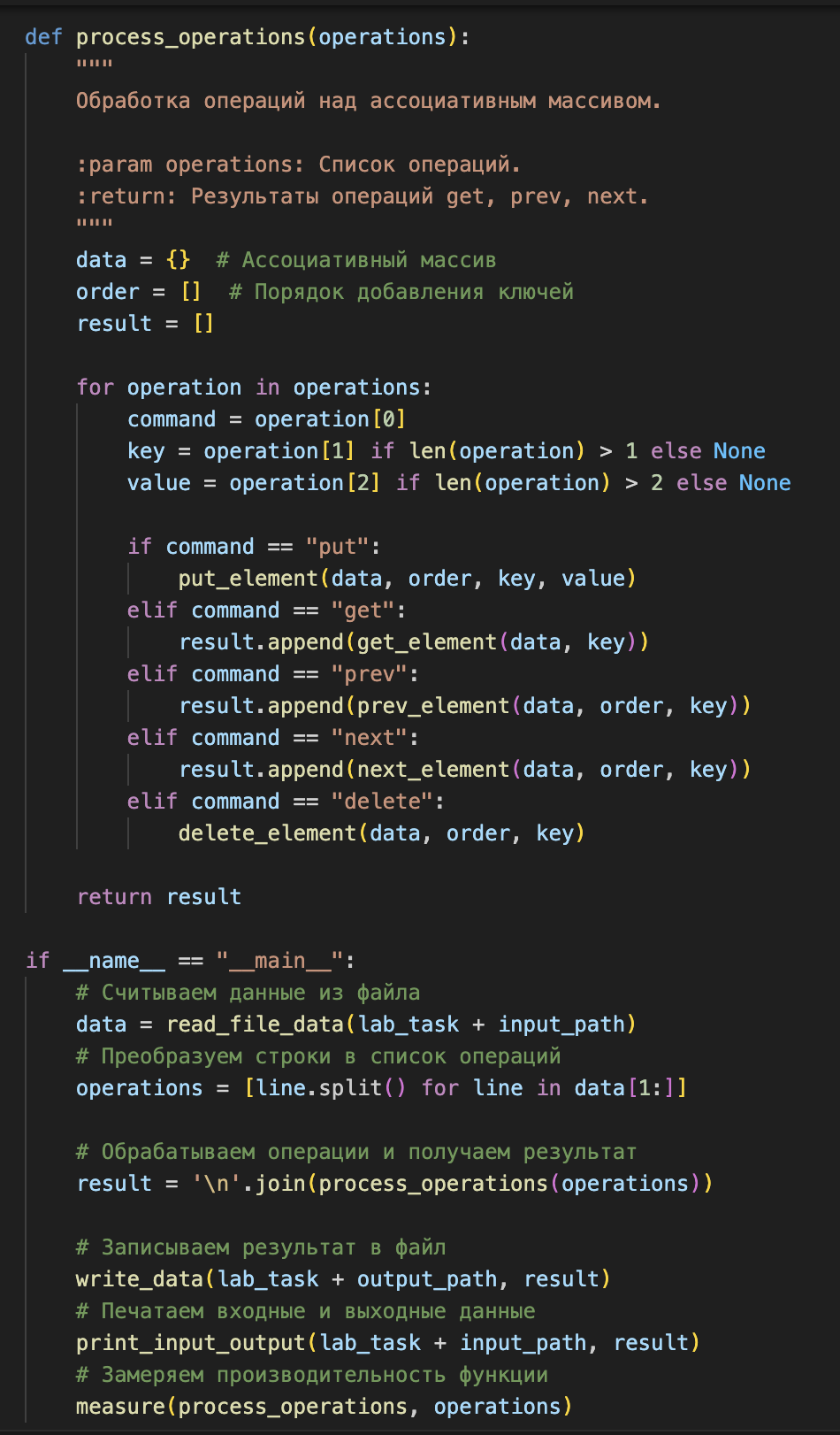
Задача №4:





Решение :





Этот код представляет собой реализацию ассоциативного массива (словаря) с поддержкой операций добавления, удаления, получения и навигации по элементам.

**Функции**

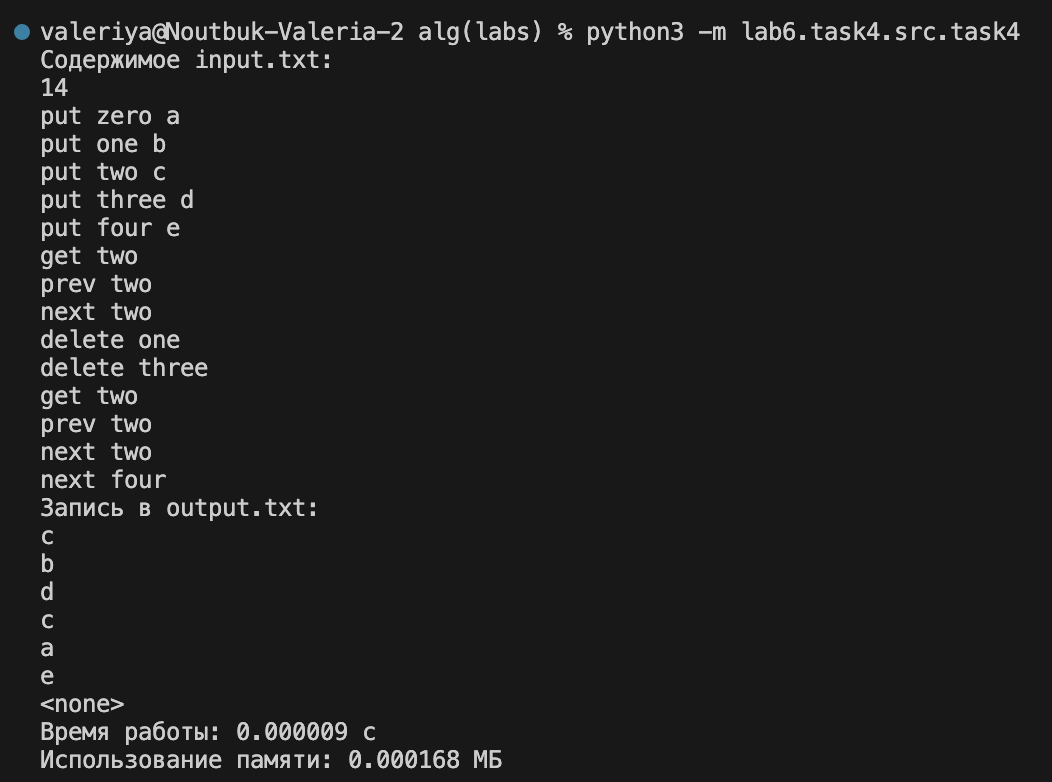
1. put\_element(data, order, key, value) - добавляет элемент в ассоциативный массив.
   * Если ключ уже существует, значение обновляется.
   * Если ключ не существует, он добавляется в массив и список порядка добавления ключей.
2. get\_element(data, key) - возвращает значение элемента по ключу.
   * Если ключ не существует, возвращает строку none.
3. prev\_element(data, order, key) - возвращает значение предыдущего элемента относительно заданного ключа.
   * Если ключ не существует, возвращает строку none.
   * Если ключ является первым в списке порядка добавления ключей, возвращает строку none.
4. next\_element(data, order, key) - возвращает значение следующего элемента относительно заданного ключа.
   * Если ключ не существует, возвращает строку none.
   * Если ключ является последним в списке порядка добавления ключей, возвращает строку none.
5. delete\_element(data, order, key) - удаляет элемент из ассоциативного массива.
   * Если ключ не существует, ничего не делает.

**Функция** process\_operations

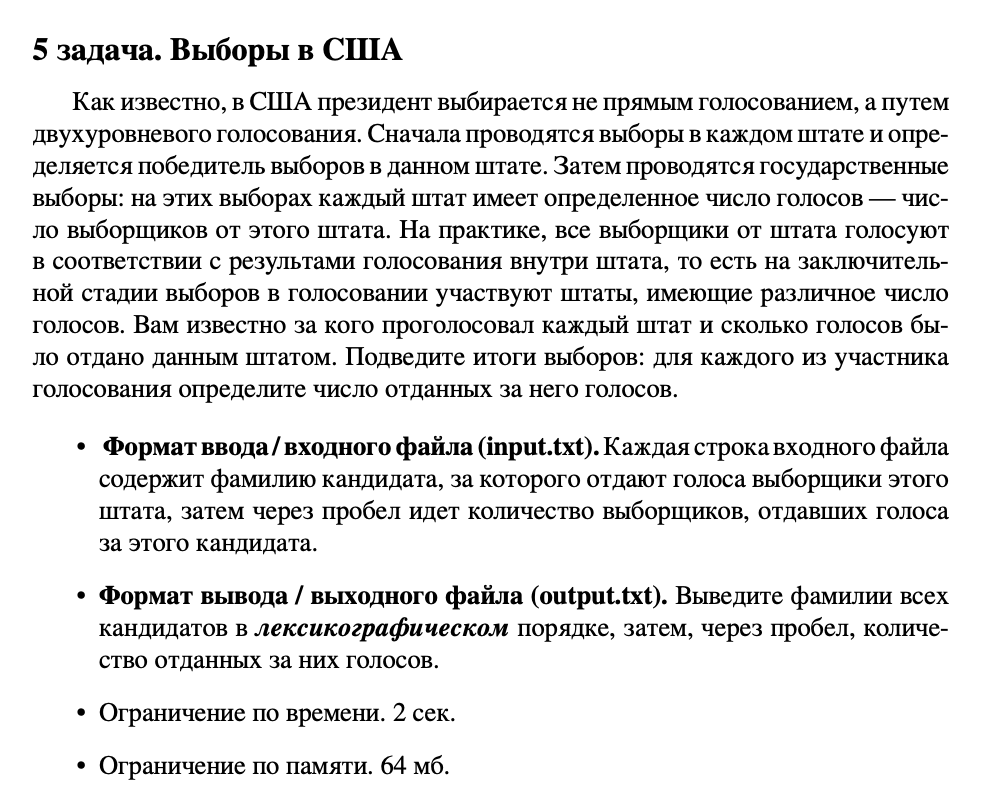
Эта функция обрабатывает список операций над ассоциативным массивом.

1. Инициализирует ассоциативный массив data и список порядка добавления ключей order.
2. Инициализирует список результатов result.
3. Обрабатывает каждую операцию в списке операций.
   * Если операция является "put", вызывает функцию put\_element.
   * Если операция является "get", вызывает функцию get\_element и добавляет результат в список результатов.
   * Если операция является "prev", вызывает функцию prev\_element и добавляет результат в список результатов.
   * Если операция является "next", вызывает функцию next\_element и добавляет результат в список результатов.
   * Если операция является "delete", вызывает функцию delete\_element.
4. Возвращает список результатов

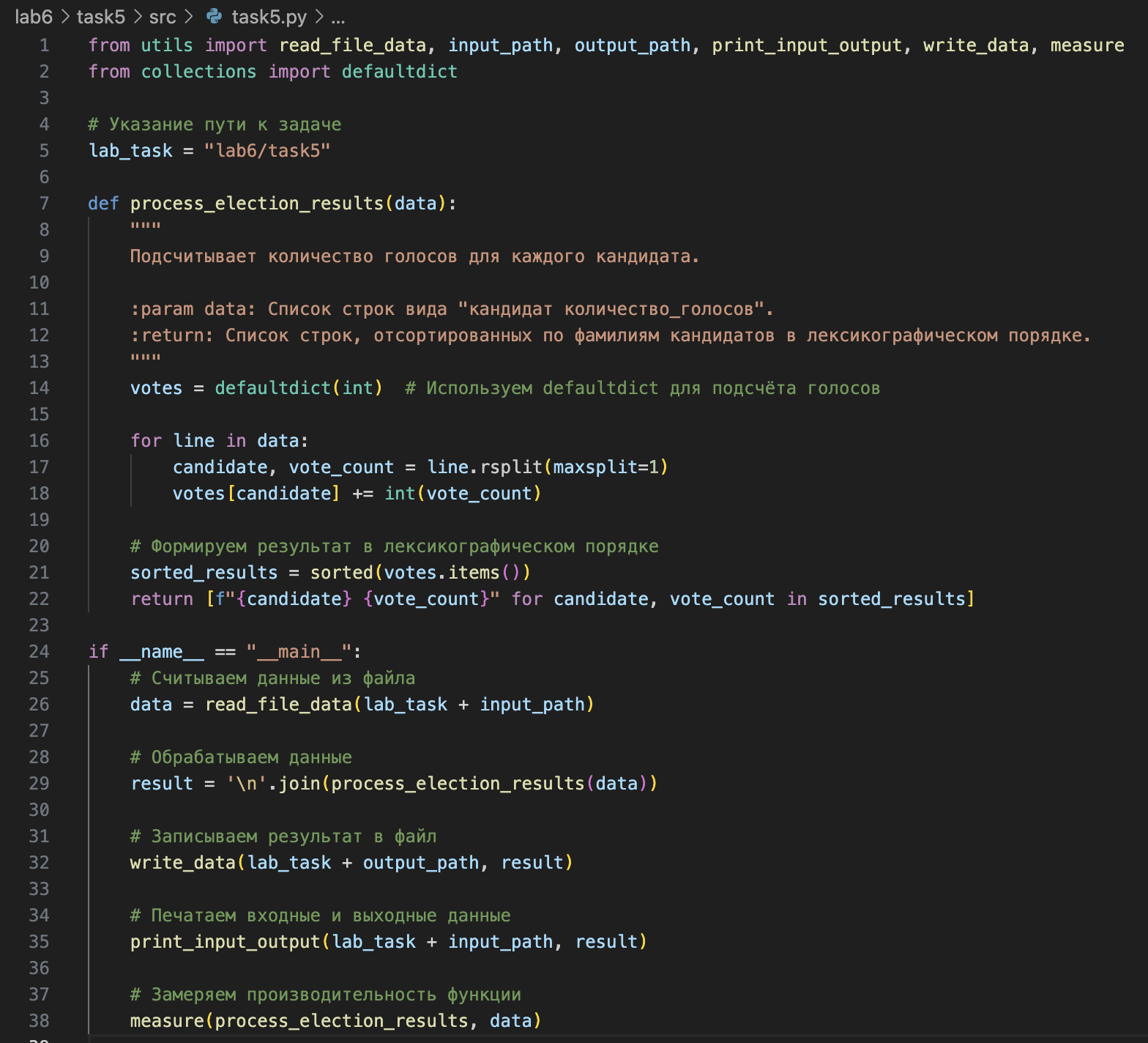
Запустила код и получила результат:



Задача №5:



Решение:



Этот код представляет собой функцию process\_election\_results, которая принимает список строк, где каждая строка представляет собой имя кандидата и количество голосов, отданных за него. Функция подсчитывает общее количество голосов для каждого кандидата и возвращает список строк, отсортированных по фамилиям кандидатов в лексикографическом порядке.

**Использование defaultdict**

В функции используется defaultdict из модуля collections, который позволяет создать словарь с заданным типом значений по умолчанию. В данном случае, если ключа нет в словаре, он будет создан с значением 0.

**Подсчет голосов**

В цикле for функция обрабатывает каждую строку в списке data. Для каждой строки она разделяет строку на имя кандидата и количество голосов с помощью метода rsplit с параметром maxsplit=1, который разделяет строку только один раз, с правой стороны.

Затем функция увеличивает значение голосов для кандидата в словаре votes на количество голосов, указанное в строке.

**Формирование результата**

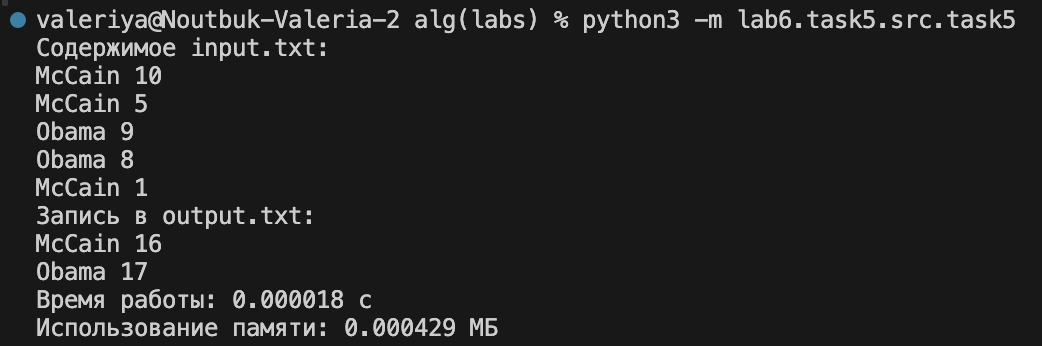
После подсчета голосов функция формирует результат в лексикографическом порядке с помощью метода sorted. Этот метод возвращает список кортежей, где каждый кортеж содержит имя кандидата и количество голосов.

Затем функция преобразует список кортежей в список строк с помощью генератора списков. Каждая строка имеет формат "кандидат количество\_голосов".

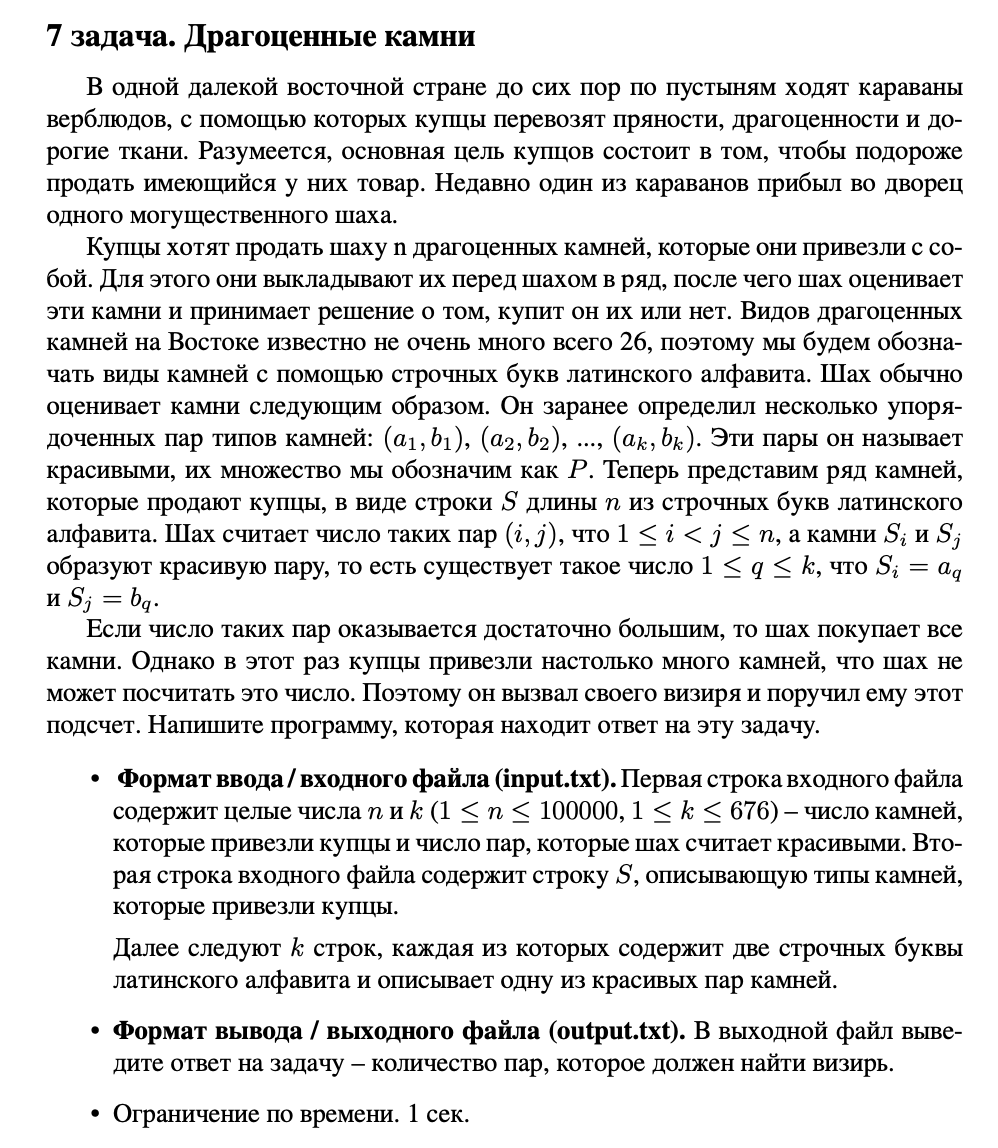
**Возвращение результата**

Наконец, функция возвращает список строк, отсортированных по фамилиям кандидатов в лексикографическом порядке.

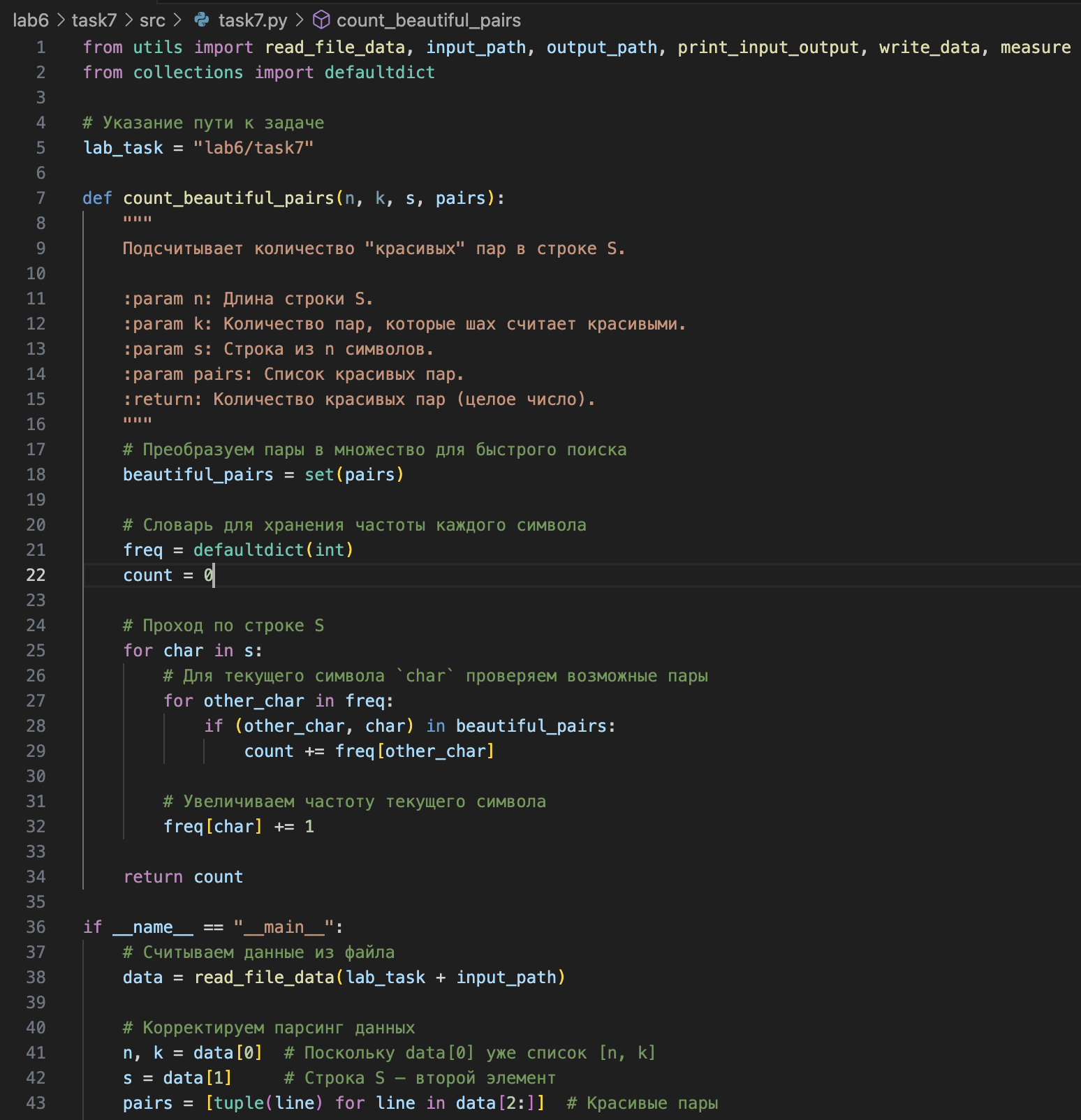
Запустила код и получила результат:

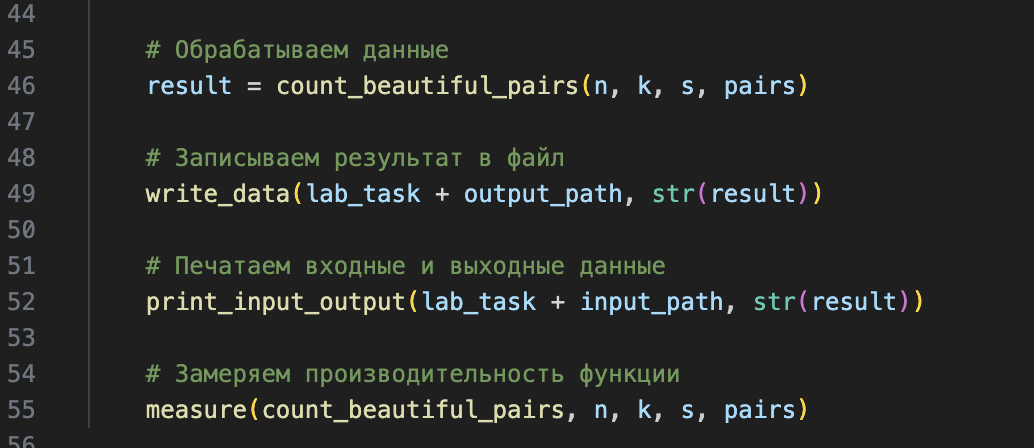


Задача №7:



Решение:





Этот код представляет собой функцию count\_beautiful\_pairs, которая принимает четыре параметра:

* n: Длина строки S.
* k: Количество пар, которые шах считает красивыми.
* s: Строка из n символов.
* pairs: Список красивых пар.

Функция возвращает количество "красивых" пар в строке S.

**Преобразование пар в множество**

Первой строкой кода является преобразование списка пар в множество beautiful\_pairs. Это делается для того, чтобы обеспечить быстрый поиск пар в дальнейшем.

**Инициализация словаря частот**

Далее инициализируется словарь freq с помощью defaultdict из модуля collections. Этот словарь будет хранить частоту каждого символа в строке S.

**Инициализация счетчика**

Переменная count инициализируется для хранения количества "красивых" пар.

**Проход по строке S**

В цикле for функция обрабатывает каждый символ char в строке S.

**Проверка возможных пар**

Для каждого символа char функция проверяет возможные пары с помощью цикла for по словарю freq. Если пара (other\_char, char) находится в множестве beautiful\_pairs, функция увеличивает счетчик count на значение freq[other\_char].

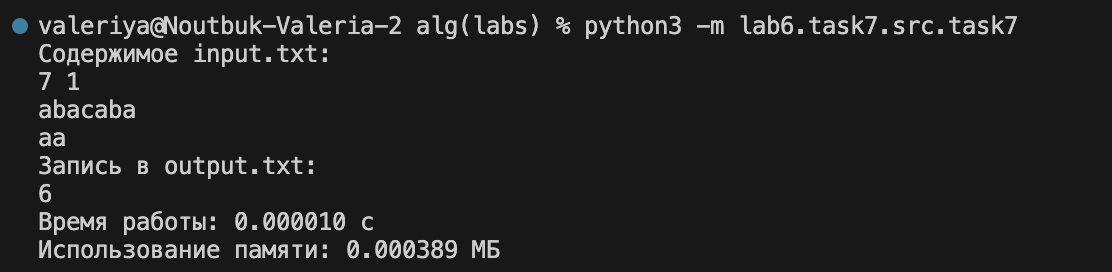
**Увеличение частоты текущего символа**

После проверки возможных пар функция увеличивает частоту текущего символа char в словаре freq на 1.

**Возвращение результата**

Наконец, функция возвращает значение счетчика count, которое представляет количество "красивых" пар в строке S.

Запустила код и получила результат:



Вывод:

Данную лабораторную полностью реализовала, а именно решила необходимое количество задач. Я отработала навыки по использованию словарей, а также получила новые знания.