A. Поисковая система

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Язык | Ограничение времени | Ограничение памяти | Ввод | Вывод |
| Все языки | 4 секунды | 128Mb | стандартный ввод или input.txt | стандартный вывод или output.txt |
| Golang 1.14.4 + network | 6 секунд | 128Mb |
| Node.js 14.15.5 | 6.5 секунд | 400Mb |
| Python 3.7.3 | 6.5 секунд | 128Mb |
| gc go | 6 секунд | 128Mb |
| Mono C# 5.2.0 | 4 секунды | 400Mb |
| Oracle Java 8 | 4 секунды | 400Mb |
| OpenJDK Java 11 | 4 секунды | 400Mb |
| Node JS 8.16 | 6.5 секунд | 400Mb |

Тимофей пишет свою поисковую систему.

Имеется n документов, каждый из которых представляет собой текст из слов. По этим документам требуется построить поисковый индекс. На вход системе будут подаваться запросы. Запрос —– некоторый набор слов. По запросу надо вывести 5 самых релевантных документов.

Релевантность документа оценивается следующим образом: для каждого уникального слова из запроса берётся число его вхождений в документ, полученные числа для всех слов из запроса суммируются. Итоговая сумма и является релевантностью документа. Чем больше сумма, тем больше документ подходит под запрос.

Сортировка документов на выдаче производится по убыванию релевантности. Если релевантности документов совпадают —– то по возрастанию их порядковых номеров в базе (то есть во входных данных).

Формат ввода

В первой строке дано натуральное число *n* —– количество документов в базе (*1 ≤ n ≤ 104*).

Далее в *n* строках даны документы по одному в строке. Каждый документ состоит из нескольких слов, слова отделяются друг от друга одним пробелом и состоят из маленьких латинских букв. Длина одного текста не превосходит *1000* символов. Текст не бывает пустым.

В следующей строке дано число запросов —– натуральное число *m* (*1 ≤ m ≤ 104*). В следующих *m* строках даны запросы по одному в строке. Каждый запрос состоит из одного или нескольких слов. Запрос не бывает пустым. Слова отделяются друг от друга одним пробелом и состоят из маленьких латинских букв. Число символов в запросе не превосходит *100*.

Формат вывода

Для каждого запроса выведите на одной строке номера пяти самых релевантных документов. Если нашлось менее пяти документов, то выведите столько, сколько нашлось. Документы с релевантностью 0 выдавать не нужно.

Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 3  i love coffee  coffee with milk and sugar  free tea for everyone  3  i like black coffee without milk  everyone loves new year  mary likes black coffee without milk | 1 2  3  2 1 |

Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 6  buy flat in moscow  rent flat in moscow  sell flat in moscow  want flat in moscow like crazy  clean flat in moscow on weekends  renovate flat in moscow  1  flat in moscow for crazy weekends | 4 5 1 2 3 |

# B. Хеш-таблица

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Язык | Ограничение времени | Ограничение памяти | Ввод | Вывод |
| Все языки | 5 секунд | 64Mb | стандартный ввод или input.txt | стандартный вывод или output.txt |
| Node.js 14.15.5 | 5 секунд | 256Mb |
| Python 3.7.3 | 15 секунд | 64Mb |
| Mono C# 5.2.0 | 5 секунд | 400Mb |
| Oracle Java 8 | 5 секунд | 400Mb |
| OpenJDK Java 11 | 5 секунд | 400Mb |
| Node JS 8.16 | 5 секунд | 256Mb |

Тимофей, как хороший руководитель, хранит информацию о зарплатах своих сотрудников в базе данных и постоянно её обновляет. Он поручил вам написать реализацию хеш-таблицы, чтобы хранить в ней базу данных с зарплатами сотрудников.

Хеш-таблица должна поддерживать следующие операции:

* put key value —– добавление пары ключ-значение. Если заданный ключ уже есть в таблице, то соответствующее ему значение обновляется.
* get key –— получение значения по ключу. Если ключа нет в таблице, то вывести «None». Иначе вывести найденное значение.
* delete key –— удаление ключа из таблицы. Если такого ключа нет, то вывести «None», иначе вывести хранимое по данному ключу значение и удалить ключ.

В таблице хранятся уникальные ключи.

Требования к реализации:

* Нельзя использовать имеющиеся в языках программирования реализации хеш-таблиц (std::unordered\_map в С++, dict в Python, HashMap в Java, и т. д.)
* Число хранимых в таблице ключей не превосходит *105*.
* Разрешать коллизии следует с помощью метода цепочек или с помощью открытой адресации.
* Все операции должны выполняться за *O(1)* в среднем.
* Поддерживать рехеширование и масштабирование хеш-таблицы не требуется.
* Ключи и значения, id сотрудников и их зарплата, —– целые числа. Поддерживать произвольные хешируемые типы не требуется.

## Формат ввода

В первой строке задано общее число запросов к таблице *n* (*1≤ n≤ 106*).

В следующих *n* строках записаны запросы, которые бывают трех видов –— get, put, delete —– как описано в условии.

Все ключи и значения –— целые неотрицательные числа, не превосходящие *109*.

## Формат вывода

На каждый запрос вида get и delete выведите ответ на него в отдельной строке.

### Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 10  get 1  put 1 10  put 2 4  get 1  get 2  delete 2  get 2  put 1 5  get 1  delete 2 | None  10  4  4  None  5  None |

### Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 8  get 9  delete 9  put 9 1  get 9  put 9 2  get 9  put 9 3  get 9  <https://go.dialektika.com/alg07_03>  https://go.dialektika.com/alg07\_04  https://go.dialektika.com/alg07\_05  https://go.dialektika.com/alg07\_06 | None  None  1  2  3 |