Номинация «ИТ-класс»
Направление «Единое»
Этап Практический

Вариант 1

Добрый день! Сегодня вам предстоит побывать в роли стажера в компанию по разработке стриминговых сервисов. В один из солнечных дней твоей работы пришла новость о том, что один из junior-разработчиков случайно стер рабочую базу данных, но у него остался бэкап в виде csv файла. Вам предстоит проявить волю к победе и восстановить минимальный функционал для работы приложения. Думаю, что Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *songs.csv,* который хранит в себе информацию о названиях песен, артистов, дат выхода и даже количество прослушиваний.

Столбцы: *streams(кол-во прослушиваний - целое число), artist_name, track_name, date(дата в формате ДД.ММ.ГГГГ).*

Разделитель *«;».*

Задача 1.

Дата обращения: 12.05.23

К нам обратилась компания, которая хочет получить все песни по дате выхода не позже 01.01.2002. Пока мы не доделали весь функционал нашей БД - найдите необходимые песни и предоставьте их в виде отчета в формате: “<Название песни> - <артист> - <кол-во прослушиваний>”.

В процессе поиска вы увидели, что не у всех песен есть кол-во прослушиваний(те равное нулю), а так сдавать отчет нельзя, поэтому перед тем как отдать отчет предоставленный выше измените данные в таблице songs.csv исходя из правила. Кол-во прослушиваний рассчитывается по формуле:

$$t_n = \left| \frac{d_n - d_i}{L_n + L_s} \right| * 10000, \text{где}$$

L_n – кол-во символов в имени артиста,

L_s – кол-во символов в названии песни,

d_n – дата указанная в начале задания,

d_i – дата выхода песни

разница дат - кол-во дней

Измененные данные запишите в таблицу songs_new.csv

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Чтобы в дальнейшем нам было удобнее работать с данными из нашей таблицы(songs.csv) отсортируйте их по столбцу дата в порядке возрастания с помощью быстрой сортировки (в задаче нельзя использовать встроенные функции сортировок!). Из полученных данных выведите топ-5 самых ранних песен. Формат каждой строки: “<№> <Название песни>, <Артист>, <дата выхода>”, где № - место в рейтинге.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей имя артиста, а на выход будет выдавать его одну любую песню, если ничего не найдено будет выводить: “К сожалению, ничего не удалось найти”. Программа должна всегда запрашивать имя. Прекратить свою работу она сможет только после ввода “0”.

Поиск необходимо осуществить с помощью алгоритма с асимптотической сложностью **O(n).**

Формат ответа на запрос пользователя: “У <артист> найдена песня: <название песни>”

Поиск необходимо осуществлять в файле songs.csv

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Компания хочет провести статистику каких исполнителей в базе данных больше - зарубежных или отечественных. Для того чтобы решить это задание, создайте два списка - один должен содержать имена исполнителей на русском языке (russian_artists), второй - на остальных (foreign_artists). Имена исполнителей относятся к исполнителям на русском языке в случае, если в названии исполнителя есть хотя бы одна русская буква. Выведите информацию о количестве элементов в каждом из созданных списков в следующем формате:

Количество российских исполнителей: {длина списка russian_artists}

Количество иностранных исполнителей: {длина списка foreign_artists}

Обратите внимание, что каждый из исполнителей должен встречаться в списке только ОДИН раз.

Запишите содержимое списков в файлы russian_artists.txt и foreign_artists.txt соответственно.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 5

Компании нужно понять, какой исполнитель в базе данных является наиболее популярным. Для решения этой задачи реализуйте хэш-таблицу, в которой ключом будет являться имя исполнителя, а значением количество его песен в базе данных. В случае, если один и тот же трек исполняется несколькими артистами, необходимо посчитать этот трек для того артиста, который указан первым. Выведите первые 10 артистов из полученной таблицы в формате

<Название исполнителя> выпустил <количество песен> песен.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 2

В далекой неизученной галактике произошла страшная катастрофа. Все корабли, бороздящие просторы космоса в этой области, затянуло в гигантскую чёрную дыру, искажающую пространство и время. В результате этого, космические судна потеряли свои изначальные координаты, и теперь их судьба была неизвестна. В черной дыре действовали совершенно другие законы физики, и выбраться из нее было практически невозможно. Но команды кораблей не сдавались, они продолжали бороться за свою жизнь, используя последние достижения науки и техники. На одном из кораблей смогли отправить файл с данными о кораблях на Землю и Вы смогли уловить сигнал и получить данные, но из-за далекого расстояния данные немного побились. Разберитесь, что было в исходном файле.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
```

imag – описание аргумента

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *space.txt,* который хранит в себе информацию о имени корабля, название родной планеты корабля, координаты последнего места связи, направление движения.

Столбцы: *ShipName(Код корабля-номер корабля), planet, coord_place(в формате x, y через пробел), direction(в формате вектора размерности два через пробел)*

Разделитель *«*».*

Кодировка: "utf-8"

Задача 1.

Самое первое чем Вам необходимо заняться это исправления побитых данных в файле *space.txt*. В столбце *coord_place* помимо истинных значений попадаются нулевые. Из штаба Вам прислали формулу по которой можно вычислить приближенные к истине значения координат последнего места связи. Выглядит она так:

$$\begin{cases} x = n + x_d & : n > 5 \\ x = -(n + x_d) * 4 + t & : n \leq 5 \\ y = m + t + y_d & : m > 3 \\ y = -(n + y_d) * m & : m \leq 3 \end{cases}$$

где n – первая цифра в номере корабля, m – вторая цифра в номере корабля, t - кол-во букв в родной планете корабля. x_d, y_d – координаты вектора направления из последнего столбца. После исправления данных запишите результат в файл *space_new.txt* и выведите все корабли, где последний элемент их кода равен "V".

Формат вывода: " <ShipName> - (<x>, <y>)", где x, y - координаты последнего места связи(уже измененные в соответствии с формулой).

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

В главном штабе обнаружили некую созависимость номера корабля и его исчезновением, поэтому вам необходимо отсортировать данные по номеру корабля в порядке возрастания и вывести 10 первых кораблей в получившемся списке.

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь сортировкой с вычислительной сложностью алгоритма — O(n2).

Формат вывода: Название корабля

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному названию корабля выводит название родной планеты и направление движения в формате:

“Корабль <ShipName> был отправлен с планеты: <*planet*> и его направление движения было: <*direction*>”

Программа должна постоянно самостоятельно запрашивать название корабля(без вопроса) и остановиться только в случае ввода ‘stop’.

В случае, если корабль будет не найден вывести: “error.. er.. ror..”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Мы готовы отправить спасательные экспедиции. Но чтобы их смогла опознать система безопасности потерявшихся кораблей, вам необходимо придумать уникальное слово-пароль. Реализуйте методы/функции, которые будут генерировать пароли для пропавших кораблей. Пароль должен состоять из 3-ех последних букв названия планеты + 2 центральных буквы кода корабля в обратном порядке + 3 буквы названия планеты, в обратном порядке. Буквы пароля должны быть все верхнего регистра.

Например, для экспедиции корабля KPWЯ-282 на планету Лотлориэн пароль должен выглядеть как “ИЭНWPТОЛ”.

“KPWЯ-282, Лотлориэн, 11 69,-5 -2” → “ KPWЯ-282;Лотлориэн;30 -51;-2 8, ИЭНWPТОЛ ”

На вход подается space.txt файл, который необходимо записать в список, для каждого элемента сгенерировать пароль, после чего дополнить список сгенерированными элементами. Последним этапом полученный список записать в новый space_uniq_password.csv файл. Назвать данный столбец “password”.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Наши ученые поняли, что поиск по номеру корабля и планете неэффективен. Вам нужно создать функцию/метод, которая будет выводить все корабли улетевшие с заданной планеты. Для решения этой задачи реализуйте хэш-таблицу, в которой значением будет являться код и номер кораблей, а ключом планета с которой произошел взлёт. Вывести информацию о 10 кораблях, использовать формат “{ключ}: {значение}”.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 3

Вы с друзья решили открыть продуктовый магазин. Друзья отвечают за рекламу и операционную деятельность. Спустя несколько месяцев работы вы поняли, что часть продуктов не распродается и ее необходимо списывать, что негативно влияет на общую прибыль. Ваша задача состоит в том, чтобы на основе накопленных данных написать функционал приложения для автоматизации учета продаж продуктов.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *products.csv,* который хранит в себе информацию о продукте, категории, цены за 1 кг, количество проданных товаров.

Столбцы: *Category, product, Date(в формате ДД.ММ.ГГГГ), Price per unit(вещественное число), Count(вещественное число)*

Разделитель *«;».*

Кодировка: "utf-8-sig"

Задача 1.

Вы собрали уже большое кол-во данных и хотите подвести итог полученной выручки за весь период. Для этого воспользуйтесь данными из таблицы *products.csv.* Все результаты по продуктам необходимо записать в новый столбец с названием total (значение расчитывать исходя из столбцов *Price per unit и Count)* и сохранить в таблицу products_new.csv. В консоль необходимо вывести итоговую сумму по категории Закуски.

Значение в столбце total рассчитывается путем умножения стоимости единицы продукции на количество проданных единиц.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортной работы и эффективной автоматизации учета продаж следует отсортировать по группам в алфавитном порядке категории товаров в таблице *products.csv* и в первой категории вывести самый дорогой товар.

Формат вывода: "В категории: <Category> самый дорогой товар: <product> его цена за единицу товара составляет <Price per unit>"

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь сортировкой вставками.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному названию категории выводит продукт с наименьшим кол-вом проданных единиц в заданной категории в формате:

“В категории: <Category> товар: <product> был куплен <Count> раз”, если по заданной категории ничего не найдено, то вывести “Такой категории не существует в нашей БД”

Программа должна постоянно самостоятельно запрашивать название категории и остановиться только в случае ввода слова ‘молоко’.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вы решили устроить акцию в магазине. Для этого вам необходимо создать промокоды для каждого товара. Реализуйте методы/функции, которые будут генерировать промокоды для продуктов. Промокод должен состоять из первых 2-ух букв названия + день поступления + 2 предпоследних буквы названия в обратном порядке + месяц поступления в обратном порядке.

Например, для товара “Напитки для здоровья” поступившего 17.08.2022 промокод должен выглядеть как “НА17ЯЬ80”.

“Напитки;Напитки для здоровья;12.06.2022;297.0;25.0” → “Напитки;Напитки для здоровья;12.06.2022;297.0;25.0;НА12ЯЬ60”

На вход подается CSV файл, который необходимо записать в список, для каждого элемента сгенерировать промокод, после чего дополнить список сгенерированными элементами, записать сгенерированные значения в столбец promocode. Последним этапом полученный список записать в новый product_promo.csv файл.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Для проведения маркетинговой кампании необходимо определить, какие из продуктов продаются меньше всего. Реализуйте хэш-таблицу, где ключом будет являться название категории продукта, а значением количество его продаж.

Выведите список из 10 наименее продаваемых продуктов в формате:

<Название категории>, <количество>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 4

В современном научном сообществе возник настоящий переизбыток информации о различных препаратах и их свойствах. Учёные из разных стран мира публикуют тысячи новых рецептов и формул, которые необходимо систематизировать и упорядочить. Это затрудняет процесс поиска необходимых данных и замедляет научный прогресс. Для решения этой проблемы, вам необходимо создать специальную программу, которая позволит быстро и эффективно находить нужные рецепты препаратов и значительно сокращать время, затрачиваемое на поиски. Эта программа станет незаменимым помощником для учёных, позволяя им сосредоточиться на более важных задачах и исследованиях.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""
if imag == 0.0 and real == 0.0: return complex_zero
...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *scientist.txt,* который хранит в себе информацию о ФИО ученого, создавшего рецепт; Название препарата; Дата создания; Ингредиенты необходимые для приготовления

Столбцы: *ScientistName, preparation, date(в формате ГГГГ-ММ-ДД), components(ингредиенты препарата через пробел)*

Разделитель *«#».*

Кодировка: "utf-8"

Задача 1

Перед началом создания программы для оптимизации поиска препаратов, необходимо удалить из списка ученых, которые выдают препарат за свой. Определить злоумышленника очень просто: оригинальный препарат имеет самую раннюю дату создания. Все что идет позже - фейк. Создайте новый файл с названием *scientist_origin.txt(данные должны быть отсортированы по дате в порядке возрастания).* Вам также необходимо срочно найти всех подельников препарата Аллопуринол, так как их рецептура неверна и из-за этого пострадало много людей. Отправьте отчет полиции в формате:

"Разработчиками Аллопуринола были такие люди (результаты выведите в порядке возрастания даты):

<ФИО ученого> - <Дата создания>

…

Оригинальный рецепт принадлежит: <ФИО ученого>"

Для решения задачи можно воспользоваться встроенными сортировками или решением задачи 2.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Чтобы поиск препаратов в файле *scientist.txt* не занимал много времени отсортируйте данные по столбцу с датой в порядке возрастания, а также выведите пять самых ранних препаратов в формате: “<ФИО автора>: <название препарата>”

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь быстрой сортировкой.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданной дате выводит ученого в формате фамилия и инициаллы и его препарат, если учёных несколько то вывести всех. Вывод сделать в формате:

“Ученый <Фамилия И.О.> создал препарат: <препарат> - <дата>”, если по заданной дате ничего не найдено, то вывести “В этот день ученые отдыхали”

Программа должна постоянно запрашивать дату и остановиться только в случае ввода слова ‘эксперимент’.

Поиск необходимо осуществить с помощью двоичного поиска в файле *scientist.txt***.**

Формат ввода даты: ДД.ММ.ГГГГ

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вам необходимо создать личные кабинеты для каждого ученого, чтобы каждый мог видеть информацию о себе, а так же иметь возможность быстрой отправки документов о новом изобретение. Для этого необходимо создать логины и пароли для каждого из ученых. Реализуйте методы/функции, которые будут генерировать логины и пароли для пользователей. Логин должен состоять из фамилии и инициалов, например, если ученого зовут Гайчеров Борис Вадимович, его логин должен выглядеть как Гайчеров_БВ. Также для каждого пользователя необходимо сгенерировать пароль, пароль должен состоять из 10 символов, включать в себя заглавные, строчные буквы английского алфавита и цифры.

“Гайчеров Борис Вадимович, Мелоксикам, 1442-08-15, Chemical_45 Chemical_24 Chemical_59” → “Гайчеров Борис Вадимович, Мелоксикам, 1442-08-15, Chemical_45 Chemical_24 Chemical_59, Гайчеров_БВ, PO891Lkov6”

На вход подается файл scientist.txt, который необходимо записать в список, для каждого элемента сгенерировать логин и пароль, после чего дополнить список сгенерированными элементами. Последним этапом полученный список записать в новый scientist_password.csv файл. Назвать эти столбцы “login” и “password”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 5

Количество новых ученых увеличивается в геометрической прогрессии, в связи с этим поиск по ФИО пользователя будет работать неэффективно. Необходимо составить хэш-таблицу, в которой будет выстроено соответствие ФИО и значения хэша ФИО. Необходимо сгенерировать хеш для каждого значения использовав представленный ниже алгоритм. Полученный хеш записать в начале строки. Назвать столбец “hash”

Для хэширования необходимо использовать следующий алгоритм.

1. Необходимо сгенерировать таблицу перестановок. В таблице находятся перемешанные числа от 0 до 1023 включительно.
2. Берем остаток от деления ASCII кода символа на 1024 – получаем индекс
3. По индексу достаем элемент из таблицы.
4. Складываем значения всех хешей и берем остаток от деления на 2048
5. Записываем значение в строку

На вход подается файл scientist.txt результаты необходимо записать в новый scientist _with_hash.csv файл.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 5

Вы встали перед выбором будущей профессии. Ваш учитель любезно предоставил вам данные по вакансиям в ведущих компаниях в формате csv файла. Вы поняли, что проанализировать данный файл вручную невозможно, после чего вам приходит идея написать функционал для более удобного понимания и представления информации.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *vacancy.csv,* который хранит в себе информацию о должностях, заработной плате, типе трудоустройства, компаниях, количестве сотрудников в компании.

Столбцы: *Salary(целое число), Work_Type, Company_Size, Role, Company.*

Разделитель *«;».*

Задача 1.

Вы поняли, что хотите сначала проанализировать какие профессии самые востребованные и высокооплачиваемые в разных компаниях. Для этого создайте таблицу vacancy_new.csv в которую запишите три столбца company, role, Salary. При этом сама вакансия(Role) должна иметь максимальный размер зарплаты в компании. После этого выведите топ-3 самых высокооплачиваемых профессий в формате: <компания> - <вакансия> - <зарплата>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для себя вы отметили, что хотели бы работать в компании с небольшим кол-вом сотрудников, так как боитесь потеряться в огромном коллективе. Поэму Вам необходимо отсортировать данные с помощью быстрой сортировки(в задаче нельзя использовать встроенные функции сортировок) по столбцу кол-во сотрудников в компании в порядке возрастания. После этого введите в консоль информацию о компании с наименьшим количеством сотрудников, где есть профессия “классный руководитель”.

Формат вывода: “В компании <Company> есть заданная профессия: <Role>, з/п в такой компании составит: <Salary>”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у

Вас название компании, а на выход будет выдавать список его вакансий, если ничего не найдено будет выводить: “К сожалению, ничего не удалось найти”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “None”.

Поиск необходимо осуществить с помощью линейного алгоритма поиска**.**

Формат ответа на запрос пользователя: “В <компании> найдена вакансия: <вакансия>. З/п составит: <Salary>”

Поиск необходимо осуществлять в файле vacancy.csv

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вы начали задумываться о том, какой процент вашей зарплаты относится к средней заработной плате на данном типе трудоустройства во всех компаниях. Реализуйте методы/функции, которые будут вычислять среднюю зарплату для определенного типа трудоустройства и определять процентное отношение данной зарплаты к средней зарплате

Пример: Средняя зарплата в типе трудоустройства:” контракт ” – 40000 руб.

(44000)/ (40000) *100% = 110%

“44000 , контракт, 35498, дизайнер взаимодействия, Avantor” → “ 44000, контракт, 35498, дизайнер взаимодействия, Avantor, 110%”

На вход подается CSV файл, который необходимо записать в список, для каждого элемента посчитать среднюю заработную плату, после чего дополнить список сгенерированными элементами, столбец необходимо назвать percent. Последним этапом полученный список записать в новый vacancy_procent.csv файл.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Чтобы легче просматривать информацию вы решили сделать группировку по названию компании. Вам необходимо вывести все вакансии данной компании в виде (Название, Зарплата, Тип трудоустройства). Для решения этой задачи реализуйте хэш-таблицу, в которой значением будет являться вакансии, а ключом название компании.

Выведите компанию с самым большим количеством вакансий.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 6

В виртуальном мире видеоигр произошла непредвиденная ситуация. Внезапный программный сбой нарушил работу всех игровых персонажей, сделав их поведение хаотичным и непредсказуемым. Игровое сообщество столкнулось с массой проблем, поскольку многие игры стали практически непроходимыми из-за возникающих ошибок и лагов. Ваша задача разработать программу, способная обнаруживать и исправлять подобные ошибки, возвращая игровым персонажам должное поведение и делая прохождение игр вновь комфортным и приятным.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *game.txt,* который хранит в себе информацию о название игры; имени персонажа, имени ошибки, дате фиксации бага.

Столбцы: *GameName, characters, nameError(буквенное значение:числовое значение), date(ГГГГ-ММ-ДД).*

Разделитель *«$».*

Задача 1.

Первая задача, которая стоит перед вами это найти все ошибки с содержанием числа 55, тк это особо важные и опасные ошибки. Для этого составьте отчет в формате: “У персонажа\t<characters>\tв игре\t<GameName>\tнашлась ошибка с кодом:\t <nameError>.\tДата фиксации:\t <date>”. После предоставления отчета измените значение ошибки на “Done”, а в поле дата поставьте “0000-00-00” и полученные измененные данные сохраните в файле game_new.csv.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

У вас снова запросили отчет по ошибкам в играх. Теперь отчет необходимо предоставить о количестве багов в каждой игре. Для этого отсортируйте данные из файла *game.txt* по столбцу игры в алфавитном порядке с помощью быстрой сортировки. После этого предоставьте отчет в формате: <Игра 1> - количество багов: <count>

….

<Игра N> - количество багов: <count>

Где N - количество уникальных игр.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у Вас имя персонажа, а на выход будет выдавать список игр, в которых встречается этот персонаж(ограничьте количество вывода элементов до 5), если ничего не найдено будет выводить: “Этого персонажа не существует”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “game”.

Поиск необходимо осуществить с помощью двоичного алгоритма поиска**.**

Обратите внимание, что данные в файле не отсортированы! Для сортировки воспользуйтесь встриоенными сортировками или напишите собственный алгоритм.

Формат ответа на запрос пользователя:

“Персонаж <characters> встречается в играх:

<Игра 1>

…

<Игра 5>

и др.”

Строчка “и др.” должна писать только в случае, если значений больше чем пять.

Поиск необходимо осуществлять в файле game.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

При просмотре логов вы увидели, что в одной игре может быть несколько ошибок. Вам необходимо добавить счетчик того, сколько ошибок выявлено в игре. Реализуйте методы/функции, которые будут считать количество упоминаний какой-либо игры. Полученный список игр необходимо отсортировать в порядке возрастания. При занесении данных в csv файл, счетчик необходимо применить ко всем строкам.

“Starfield, Mila, MBKJ:537, 1233-11-19” → “ Starfield, Mila, MBKJ:537, 1233-11-19, 9”

На вход подается файл game.txt, который необходимо записать в список, добавить счетчик для каждого элемента, после чего дополнить список счетчиками. Последним этапом полученный список записать в новый game_counter.csv файл. Назвать новый столбец “counter”.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Когда мы начали исправлять ошибки, оказалось, что поиск по названию игры и персонажа неэффективен. Необходимо составить хэш-таблицу, в которой будет выстроено соответствие название игры + имя персонажа и значения их хэша. На основании этого необходимо составить хэш-таблицу, добавить полученный хэш нулевым столбом, результат записать в csv файл.

Строка для хэша генерируется следующим образом: Название игры + Имя персонажа

Примечание: в названии игры необходимо убрать все пробелы

Название игры: Starfield Персонаж: Mila → StarfieldMila

Для хэширования необходимо использовать следующий алгоритм.

где p и m - некоторые выбранные положительные числа.

n – кол-во символов в строке

Рекомендации по выбору чисел p и m.

Для данного задания рекомендуется взять p = 65.

26*2 – кол-во букв в английском алфавите строчных и прописных

10 – кол-во цифр.

2 – символы “:”,”-“, которые могут встретиться в названии игры.

1 – т.к. считаем не с 0.

m = 109+9

Для вычисления хэша строки s, которая содержит только строчные буквы необходимо преобразовать каждый символ строки s в целое число. Можно использовать преобразование a →1, b →2, … z → 26 . Преобразование a → 0 не является хорошей идеей, поскольку тогда хэши строк a, aa, aaa, … все оцениваются как 0.

Для Заглавных буква следует использовать преобразование A → 27, B → 28 … Z → 52

Для цифр и доп символов. 1 → 53 … 0 → 63, : → 64, - → 65.

На вход подается файл game.txt результаты необходимо записать в новый game _with_hash.csv файл.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 7

Образованному человеку достаточно прочитать десять книг, но чтобы узнать, что это за книги необходимо прочитать несколько тысяч. Но мы живем в эпоху цифровой трансформации, в которую можно найти огромное количество информации. У вас есть csv файл с данными о книгах и их рейтинге, хранящихся в школьной библиотеке. Вам необходимо написать функционал для автоматизации поиска и подбора книг для чтения.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *books.csv,* который хранит в себе информацию о авторе, названии, дате выпуска, цене, рейтинге, ISBN

Столбцы: *book_id, isbn, authors, original_publication_year, original_title, ratings_1(вещественное число).*

Разделитель *«;».*

Задача 1.

Вам посоветовали прочитать книги Дж.К. Роулинг(книги с соавторством также считаются). Вы не хотите разочароваться в авторе после прочтения первой книги, поэтому приняли решение создать отдельный файл books_rowling.csv со всеми книгами данного автора (в файле должны быть столбцы: *book_id, authors, original_title, ratings_1).* После этого вам необходимо получить все книги с рейтингом больше 8. Формат вывода книг: “<Author> - <original_title>\t <ratings_1>”

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Теперь когда вы владеете документом со всеми книгами и их рейтингом, Вам хочется попробовать прочитать три книги с худшим рейтингом, чтобы понять а из-за чего им поставили такие плохие оценки. Для этого вам необходимо отсортировать документ по столбцу с рейтингом с помощью сортировки, которая имеет среднюю сложность ***O(n log n)***. После сортировки выведите три книги с худшим рейтингом в формате:

“<original_title> - <authors> - <ratings_1>”

В задаче нельзя использовать встроенные функции сортировок.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с вашим файлом. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у вас название книги, а на выход будет выдавать список книг и их идентификационные номера, если ничего не найдено будет выводить: “Данной книги в этой библиотеке нет”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “хватит”.

Поиск необходимо осуществить с помощью последовательного алгоритма поиска**.**

Формат ответа на запрос пользователя:

“<название книги> - <isbn> - <book_id> - <ratings_1>”

Строчка “и др.” должна писать только в случае, если значений больше чем пять.

Поиск необходимо осуществлять в файле books.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вам необходимо реализовать систему рекомендаций. Система будет основываться на рейтинге книг и иметь 3 ярлыка: «не рекомендовать», «рекомендовать после», «рекомендовать в первую очередь».

- «не рекомендовать»: рейтинг <5
- «рекомендовать после»: 5 <= рейтинг <8
- «рекомендовать в первую очередь» рейтинг> = 8

Реализуйте методы/функции, которые будут выставлять ярлык-оценку книге основываясь на рейтинге произведения.

“6310, 142403881, Роальд Даль Квентин Блейк, 1964, Чарли и шоколадная фабрика, 6.17” → 6310, 142403881, Роальд Даль Квентин Блейк, 1964, Чарли и шоколадная фабрика, 6.17, рекомендовать после”

На вход подается CSV файл, который необходимо записать в список, для каждого элемента сгенерировать ярлык, после чего дополнить список сгенерированными элементами. Последним этапом полученный список записать в новый books_grade.csv файл.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вы решили реализовать поиск по автору, чтоб быстро выводить его рейтинг. Для решения этой задачи реализуйте хэш-таблицу, в которой значением будет средний рейтинг книг автора, а ключом автор.

Выведите 10 авторов с самым большим средним рейтингом в формате:

Автор: <Имя автора>, Средний рейтинг: <Средний рейтинг>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 8

В одной далекой галактической системе был пойман загадочный сигнал, исходящий от неизвестной инопланетной цивилизации. После тщательного анализа сигнала, ученые обнаружили, что он содержит сложные алгоритмы и инструкции для создания сверхмощного космического корабля - ракеты. Этот корабль должен был обладать уникальными возможностями и стать настоящим прорывом в исследовании космического пространства. Для расшифровки сигнала и создания ракеты Вам необходимо разработать специальную программу, которая поможет преобразовать инопланетные коды в понятные для человека инструкции и схемы, которые затем можно будет использовать для сборки ракеты.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *rocket.txt,* который хранит в себе информацию о дате сигнала; коде сигнала; части ракеты

Столбцы: *date(ГГГГ-ММ-ДД), code, rocketparts*

Разделитель *«@».*

Задача 1.

Ввод: стандартный ввод

Вывод: стандартный вывод

Первая идея, которая пришла вам в голову это собрать уникальные запчасти и подсчитать количество шифров к ним, чтобы в дальнейшем начать дешифровку по запчастям в которых есть больше всего информации. Для этого создайте отдельный файл rocket_part.txt с разделителем @ и столбцами название части ракеты(rocketPart), кол-во шифров(countCode). После этого напишите небольшую программу, на вход которой подается название части ракеты, а ны выход кол-ва шифров.

Формат ввода: строка(гарантируется, что вводимая строка находится в файле)

Формат вывод: одно число

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для дальнейшей работы вам необходимо отсортировать ошибки в алфавитном порядке, после этого выведите все ошибки в которых содержится число 312. Так как вы предполагаете, что ошибки с таким значением содержат в себе очень много полезной информации.

Формат вывода: “<code> для <rocketpart>”

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь сортировкой вставками.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному названию шифра будет выводить дату получение сигнала, а также от какой части ракеты этот шифр в формате:

“Шифр: <code> от: <rocketpart> был получен <date>”, если по заданной категории ничего не найдено, то вывести “такой сигнал еще не был получен”

Программа должна постоянно самостоятельно запрашивать название категории и остановиться только в случае ввода слова ‘nlo’.

Поиск необходимо осуществить с помощью линейного алгоритма поиска**.**

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Нашим ученым необходимо понять с какого элемента начать сборку. Вам необходимо сгенерировать уникальные номера для каждого элемента. Реализуйте методы/функции, которые будут генерировать уникальные номера для компонентов. Номер должен состоять из порядка использования данной детали (смотреть по дате получения запроса) + код сигнала + первая буква из каждого слова названия. Уточнение: порядок использования детали зависит от даты её получения, чем раньше пришла деталь, тем меньше её номер в порядке.

Для удобства вычисления рекомендуется отсортировать массив по возрастанию даты.

“0900-03-20, НБПК464, Ускоритель” → “0900-03-20, НБПК464, Ускоритель, 1НБПК464У”

“0931-11-19, ГПFA309, Батарея электронных блоков” → “0931-11-19, ГПFA309, Батарея электронных блоков, 12ГПFA309БЭБ”

На вход подается файл rocket.txt, который необходимо записать в список, добавить уникальный код для каждого элемента, после чего дополнить список сгенерированным кодом. Последним этапом полученный список (отсортированный по возрастанию даты) записать в новый rocket_code.csv файл. Новый столбец назвать “queue”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Инженеры запросили 10 самых часто используемых в сборке деталей. Для решения этой задачи реализуйте хэш-таблицу, в которой значением будет являться название детали, а ключом количество необходимое в ракете. В случае равного количества деталей брать ту, о которой раньше пришел сигнал.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 9

В далекой неизученной галактике произошла страшная катастрофа. Все корабли, бороздящие просторы космоса в этой области, затянуло в гигантскую чёрную дыру, искажающую пространство и время. В результате этого, космические судна потеряли свои изначальные координаты, и теперь их судьба была неизвестна. В черной дыре действовали совершенно другие законы физики, и выбраться из нее было практически невозможно. Но команды кораблей не сдавались, они продолжали бороться за свою жизнь, используя последние достижения науки и техники. На одном из кораблей смогли отправить файл с данными о кораблях на Землю и Вы смогли уловить сигнал и получить данные, но из-за далекого расстояния данные немного побились. Разберитесь, что было в исходном файле.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
```

```
imag – описание аргумента

"""
if imag == 0.0 and real == 0.0: return complex_zero
...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *space.csv,* который хранит в себе информацию о имени корабля, название родной планеты корабля, координаты последнего места связи, направление движения.

Столбцы: *ShipName(Код корабля-номер корабля), planet, coord_place(в формате x, y - целочисленные, через пробел), direction(в формате вектора размерности два через пробел, числа - целочисленные)*

Разделитель *«,».*

Кодировка: "utf-8"

Задача 1.

Пока мы принимали сигнал возникали помехи и некоторые данные не записались. В столбце *coord_place* наибольшее число побитых данных, те тех где вместо координат лежат нули. Пожалуйста, составьте отчет для штаба и вышлите его в файле *space_new.txt.*

Формат строк в отчете: "При получении данных о корабле <*ShipName*> возникли сбои. Предположительные координаты - S_x, S_y."

Где $S_x = N - direction_x$, $S_y = N - direction_y$, а N - количество символов в названии родной планеты, $direction_x$ и $direction_y$ - координаты направления соответственно.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Через какое-то время было принято сложное решение отправить поисковый корабль по координатам самым близким к Земле. Для этого отсортируйте столбец *coord_place* в порядке возрастания по расстоянию от последнего места связи и Земли. Расчеты производить исходя из того, что координаты Земли: (0, 0).

Формула расчета расстояния:

$$\rho = \sqrt{(coord_x - Earth_x)^2 + (coord_y - Earth_y)^2}$$

Также выведите тройку кораблей у которых координаты не побиты, те не равны 0.

Формат вывода: “Корабль: <*ShipName*> последний раз был на связи по координатам: <*coord_place*>; и двигался в направлении: <*direction*>.”

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь **быстрой сортировкой**.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному названию корабля выводит его расстояние до Земли по последнему месту связи:

“Корабль <ShipName> последний раз был на связи по координатам: <*coord_place*>, что составляет: <ρ> космических единиц.”

Формулу для расчета расстояния возьмите из предыдущей задачи. Результаты округлите до 3х знаков после запятой.

Программа должна постоянно самостоятельно запрашивать название корабля(без вопроса) и остановиться только в случае ввода ‘stop’.

В случае, если корабль будет не найден вывести: “error.. er.. ror..”

Поиск необходимо осуществить с помощью линейного алгоритма поиска**.**

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Для того, чтобы связаться с экипажами кораблей, было принято решение использовать Азбуку Морзе. Реализуйте методы/функции, которые будут генерировать сообщения для пропавших кораблей. Сообщение должно состоять из кода корабля + его координат.

Азбука Морзе имеет следующие значения:

Русский символ	Код Морзе	Английский символ	Код Морзе	Символ	Код Морзе
А	.-	A	.-	1	.----
Б	-...	B	-...	2	..---
В	.--	C	-.-.	3	...--
Г	--.	D	-..	4	....-
Д	-..	E	.	5	.....
Е, Ё	.	F	..-.	6	-....
Ж	...-	G	--.	7	--...
З	--..	H	....	8	---..
И	..	I	..	9	----.
Й	.---	J	.---	0	-----
К	-.-	K	-.-	Точка	.-.-.-
Л	.-..	L	.-..	Запятая	--..--
М	--	M	--	Двоеточие	---...
Н	-.	N	-.	;	-.-.-.
О	---	O	---	(	-.--.
П	.--.	P	.--.	)	-.--.-
Р	.-.	Q	--.-	Апостроф	.----.
С	...	R	.-.	Кавычки	.-..-.
Т	-	S	...	-	-....-
У	..-	T	-	/	-..-.
Ф	..-.	U	..-	?	..--..
Х	....	V	...-	!	-.-.--
Ц	-.-.	W	.--	Пробел	/
Ч	---.	X	-..-		
Ш	----	Y	-.--		
Щ	--.-	Z	--..		
Ь	-..-				
Ы	-.--				
Ъ	--.--				
Э	..-..				
Ю	..--				
Я	.-.-				

Азбука Морзе представлена в файле morse_code.txt.

На вход подается CVS файл, который необходимо записать в список, для каждого кода корабля сгенерировать сообщение, после чего дополнить список сгенерированными элементами. Получившейся результат необходимо записать в новый файл message_for_ship.csv.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Файл "space.csv" содержит информацию о космических кораблях. Ваша цель - создать хеш-таблицу, где ключом будет являться название планеты, а значием количество кораблей, которые были отправлены с этой планеты.

1. Прочитайте данные из файла "space.csv" и загрузите их в память вашей программы.
2. Определите хеш-функцию, которая будет преобразовывать название корабля в индекс хеш-таблицы.
3. Создайте массив (список) для хранения данных о кораблях. В каждом элементе массива будет храниться список (цепочка), который будет использоваться для размещения кораблей с одинаковым хешем.
4. Выведите 5 планет с наибольшим количеством кораблей.

Выведите информацию о корабле, если он был найден, в противном случае, выведите сообщение: "Корабль не найден".

Обратите внимание на производительность вашей хеш-таблицы и ее способность обрабатывать коллизии без потери данных.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта

2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 10

В современном научном сообществе возник настоящий переизбыток информации о различных препаратах и их свойствах. Учёные из разных стран мира публикуют тысячи новых рецептов и формул, которые необходимо систематизировать и упорядочить. Это затрудняет процесс поиска необходимых данных и замедляет научный прогресс. Для решения этой проблемы, вам необходимо создать специальную программу, которая позволит быстро и эффективно находить нужные рецепты препаратов и значительно сокращать время, затрачиваемое на поиски. Эта программа станет незаменимым помощником для учёных, позволяя им сосредоточиться на более важных задачах и исследованиях.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *scientist.csv,* который хранит в себе информацию о ФИО ученого, создавшего рецепт; Название препарата; Дата создания; Ингредиенты необходимые для приготовления

Столбцы: *ScientistName, preparationt, date(в формате ГГГГ-ММ-ДД), components(ингредиенты препарата через пробел)*

Разделитель *«,».*

Кодировка: "utf-8"

Задача 1

Перед созданием программы Вам захотелось собрать небольшую статистику по наиболее продуктивным месяцам для ученых. Для этого создайте новый файл *scientist_new.csv* в котором будут столбцы: month(формат месяца - словом(например, январь) и количество препаратов созданных в этом месяце(count). После этого выведите самый популярный месяц в котором было создано самое большое количество препаратов.

Формат вывода: "<Месяц> наиболее благоприятен для ученых. В этом месяце было создано - <count> препарат(-а)"

Месяц с заглавной буквы!

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Чтобы поиск препаратов не занимал много времени отсортируйте данные по столбцу с названиями препаратов в алфавитном порядке, а также выведите три самых первых перпарата в формате: "<ФИО автора>: <название препарата>"

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь сортировкой слияния.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному препарату выводит первого ученого в списке в формате фамилия и инициалы и дата создания препарата в формате - ДД.ММ.ГГГГ.

Вывод сделать в формате:

“Ученый <Фамилия И.О.> создал препарат: <дата>”, если по заданному препарату ничего не найдено, то вывести “Этот препарат еще не создан”

Программа должна постоянно запрашивать дату и остановиться только в случае ввода слова ‘эксперимент’.

Формат ввода даты: ДД.ММ.ГГГГ

Поиск необходимо осуществить с помощью алгоритма с асимптотической сложностью **O(n).**

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вам необходимо создать уникальные идентификаторы для всех имеющихся препаратов. Реализуйте методы/функции, которые буду генерировать идентификатор. Идентификатор строится по следующему формату: Первые 3 буквы Фамилии ученого (первая буква – заглавная) + первые 3 буквы названия препарата (первая буква – заглавная) + год создания.

На вход подается CSV файл, который необходимо записать в список, для каждого препарата сгенерировать идентификатор, после чего дополнить список сгенерированными элементами. Последним этапом необходимо полученный список записать в новый файл preparationt_id.csv.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Ваша задача - создать программу для управления базой данных рецептов препаратов с использованием метода закрытой адресации.

Программа должна предоставлять функциональность для добавления новых рецептов. Вы также должны обеспечить уникальность записей на основе имени препарата и даты создания.

1. Создайте хеш-таблицу для хранения информации о рецептах препаратов, где ключом будет являться название препарата + дата создания, а значением ФИО ученого, создавшего рецепт; Ингредиенты необходимые для приготовления.
2. Реализуйте функцию поиска по хеш-таблице.
3. Реализуйте функцию для добавления нового рецепта в список.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 11

В виртуальном мире видеоигр произошла непредвиденная ситуация. Внезапный программный сбой нарушил работу всех игровых персонажей, сделав их поведение хаотичным и непредсказуемым. Игровое сообщество столкнулось с массой проблем, поскольку многие игры стали практически непроходимыми из-за возникающих ошибок и лагов. Ваша задача разработать программу, способная обнаруживать и исправлять подобные ошибки, возвращая игровым персонажам должное поведение и делая прохождение игр вновь комфортным и приятным.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *game.csv,* который хранит в себе информацию о название игры; имени персонажа, имени ошибки, дате фиксации бага.

Столбцы: *GameName, characters, nameError(буквенное значение:числовое значение), date(ГГГГ-ММ-ДД).*

Разделитель *«,».*

Задача 1.

Самым важным персонажем для нашего игрового мира является Avery. Найдите все игры в которых есть данный персонаж и составьте отчет в виде файла: game_new.txt, где каждая строка будет представлена в формате: “У персонажа <characters> в игре <GameName> нашлась ошибка с кодом: <nameError>. Дата фиксации: <date>”. После предоставления отчета выведите первую игру в списке с Avery.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

У вас снова запросили отчет по ошибкам в играх. Теперь отчет необходимо предоставить о количестве багов для каждого персонажа. Для этого отсортируйте данные из файла *game.csv* по столбцу персонаж в алфавитном порядке с помощью быстрой сортировки. После этого предоставьте отчет в формате:
<Персонаж 1> - количество багов: <count>

….

<Персонаж N> - количество багов: <count>

Где N - количество уникальных персонажей.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у Вас имя ошибки, а на выход будет выдавать список игр, в которых встречается эта ошибка, если ничего не найдено будет выводить: “Этой ошибки не существует”. Программа должна всегда запрашивать название ошибки. Прекратить свою работу она сможет только после ввода “game”.

Поиск необходимо осуществить с помощью последовательного алгоритма поиска**.**

Формат ответа на запрос пользователя:

“Ошибка <nameError> встречается в игре <GameName> у персонажа <characters>.”

Поиск необходимо осуществлять в файле game.csv

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Необходимо собрать статистику по играм. Сруппируйте данные по названию игры и сделайте подсчет количества ошибок для каждой из них. Необходимо реализовать вывод списка игр с указанием суммарного количества ошибок и списком персонажей с их ошибками по следующей структуре:

```
Game 1:
  Total errors: X
  Character 1:
    - Error 1
    - Error 2
  Character 2:
    - Error 1
  ...
Game 2:
  Total errors: Y
  Character 1:
    - Error 1
  ...
```

Пример работы программы:

Предположим, что файл game.csv имеет следующее содержимое:

Game1,Character1,Error1,2022-01-01

Game1,Character1,Error2,2022-01-10

Game1,Character2,Error1,2022-01-15

Game2,Character1,Error1,2022-01-20

Результат работы программы:

```
Game 1:
  Total errors: 3
  Character 1:
    - Error 1
    - Error 2
  Character 2:
    - Error 1

Game 2:
  Total errors: 1
  Character 1:
    - Error 1
```

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вы получили файл game.csv, который содержит информацию о названии игры, имени персонажа, имени ошибки и дате фиксации бага. Для решения этой задачи реализуйте хэш-таблицу, в которой значением будет являться название игры и имя персонажа, а ключом имя ошибки. Затем программа должна анализировать данные и предоставлять следующую информацию:

1. Список всех игр, в которых были обнаружены ошибки.
2. Список всех ошибок, отсортированный по дате их фиксации.

Требования:

1. Используйте хеш-таблицы для эффективного хранения и поиска данных.
2. Программа должна выводить информацию в соответствии с описанными пунктами.

Пример содержимого файла game.csv:

Game,Character,Error,Date

Game1,Character1,Error1,2022-01-01

Game1,Character2,Error2,2022-01-02

Game1,Character1,Error3,2022-01-03

Game2,Character3,Error4,2022-01-01

Game2,Character3,Error5,2022-01-04

Game3,Character4,Error6,2022-01-05

Ожидаемый вывод программы:

Игры с ошибками:

Game1

Game2

Game3

Список ошибок по датам:

2022-01-01: Error1, Error4

2022-01-02: Error2

2022-01-03: Error3

2022-01-04: Error5

2022-01-05: Error6

Полученный список ошибок запишите в новый файл error.txt в формате:

date: ['nameError']

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта

2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 12

В одной далекой галактической системе был пойман загадочный сигнал, исходящий от неизвестной инопланетной цивилизации. После тщательного анализа сигнала, ученые обнаружили, что он содержит сложные алгоритмы и инструкции для создания сверхмощного космического корабля - ракеты. Этот корабль должен был обладать уникальными возможностями и стать настоящим прорывом в исследовании космического пространства. Для расшифровки сигнала и создания ракеты Вам необходимо разработать специальную программа, которая поможет преобразовать инопланетные коды в понятные для человека инструкции и схемы, которые затем можно будет использовать для сборки ракеты.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *rocket.csv,* который хранит в себе информацию о дате сигнала; коде сигнала; части ракеты

Столбцы: *date(ГГГГ-ММ-ДД), code, rocketparts*

Разделитель *«,».*

Задача 1.

Ввод: стандартный ввод

Вывод: стандартный вывод

Первая зацепка, которую вы получили это, то что каждый шифр связан с номером месяца, когда был получен сигнал. Составьте файл с номерами месяцев и количества полученных шифров в них. Файл назовите rocket_part.txt и строки запишите в формате: “В <номер месяца> было получено - <количество шифров> шифров”. После этого напишите небольшую программу, на вход которой подается номер месяца, а на выход кол-ва шифров в том же формате, что хранится в файле.

Формат ввода: строка(гарантируется, что вводимая строка находится в файле)

Формат вывод: строка (В <номер месяца> было получено - <количество шифров> шифров)

Пример:

Ввод: 01

Вывод: В 01 было получено - 56 шифров

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для дальнейшей работы вам необходимо отсортировать полученные сигналы по дате в порядке возрастания. После этого выведите три самых ранних сигнала, которые были получены.

Формат вывода: “Сигнал с шифром <code> был получен <date - формат ДД.ММ.ГГГГ> и предназначается для <rocketpart>”

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь сортировкой вставками.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданной дате будет выводить шифр, который был получен в этот день, а также от какой части ракеты этот шифр в формате:

“Шифр: <code> от: <rocketpart> был получен <date>”, если по заданной категории ничего не найдено, то вывести “в этот день космос молчал”

Программа должна постоянно самостоятельно запрашивать название категории и остановиться только в случае ввода слова ‘sleep’.

Поиск необходимо осуществить с помощью двоичного поиска**.**

Формат ввода даты: ГГГГ-ММ-ДД

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

При сборке ракеты выяснилось, что для настройки каждого из модулей необходимо ввести определённую команду, которая состоит из: первой буквы каждого слова в названии модуля + преобразованный код + разность годов дат. Все значения должны быть разделены пробелами.

Преобразованный код – это код, в котором сначала идут цифры, затем латинский буквы, затем русские буквы. Разность дат – это разность значений годов: текущий год – год из значения даты в столбце. Команда должна быть написана заглавными буквами.

Например, у вас есть запись:

1651-08-17 ЫQРЛ704 Регулятор тяги двигателя

Результатом будет следующее значение:

РТД 704QЫРЛ 372

На вход подается CSV файл, который необходимо записать в список, для каждого элемента сгенерировать команду, после чего дополнить список сгенерированными элементами. Последним этапом полученный список записать в новый файл rocket_commands.csv.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Для хранения и управления информацией о частях космической ракеты, содержащейся в инопланетном сигнале, требуется создать программу, использующую хеш-таблицу с открытой адресацией. Эта система будет предоставлять функциональность для добавления и обновления данных о частях ракеты.

1. Создайте структуру данных для хранения информации о частях космической ракеты.

2. Реализуйте хеш-таблицу с открытой адресацией для хранения данных о частях ракеты. Каждая часть должна быть хеширована в соответствии с ее кодом, и в случае коллизии, используйте метод открытой адресации для разрешения коллизий.

3. Разработайте функции для добавления новых частей ракеты в хеш-таблицу и для обновления информации о существующих частях ракеты.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 13

Добрый день! Сегодня вам предстоит побывать в роли стажера в компанию по разработке стриминговых сервисов. В один из солнечных дней твоей работы пришла новость о том, что один из junior-разработчиков случайно стер рабочую базу данных, но у него остался бэкап в виде csv файла. Вам предстоит проявить волю к победе и восстановить минимальный функционал для работы приложения. Думаю, что Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *songs.txt,* который хранит в себе информацию о названиях песен, артистов, дат выхода и даже количество прослушиваний.

Столбцы: *streams(кол-во прослушиваний - целое число), artist_name, track_name, date(дата в формате ДД.ММ.ГГГГ).*

Разделитель *««?».*

Задача 1.

Формат ввода: стандартный ввод

Вывод: файл songs_artst.csv

Клиенты нашей компании хотят получить доступ к песням определенных исполнителей. Для этого вам выдали задачу написать программу, на вход в которую будет подаваться имя артиста, а на выход вы будете создавать файл *songs_artst.csv* со столбцами *track_name, streams, date.*

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Чтобы в дальнейшем нам было удобнее работать с данными из нашего файла(songs.txt) отсортируйте их по имени артиста в алфавитном порядке с помощью сортировки со сложностью $O(N^2)$(в задаче нельзя использовать встроенные функции сортировок!). Из полученных данных выведите топ-5 первых песен. Формат каждой строки: “<Название песни>, <Артист>, <дата выхода>”.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей название песни, а на выход будет выдавать автора этой песни, если ничего не найдено

будет выводить: “К сожалению, ничего не удалось найти”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “0”.

Формат ответа на запрос пользователя: “Песня <*track_name*> принадлежит <*artist_name*>”

Поиск необходимо осуществить с помощью двоичного поиска**.**

Поиск необходимо осуществлять в файле songs.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Найдите всех артистов, чьи песни были выпущены до 1990 года, и выведите их список. Посчитайте среднее количество прослушиваний песен для каждого артиста и выведите результат в порядке убывания среднего количества прослушиваний.

Последним этапом необходимо записать полученный список в новый файл – songs_average.txt. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вам предстоит создать программу, которая будет обрабатывать данные из файла songs.txt, содержащего информацию о названиях песен, артистах, датах выхода и количестве прослушиваний. Для обработки данных используйте хеш-таблицы.

Вам необходимо:

Прочитать данные из файла songs.txt и создать хеш-таблицу, где ключом будет название песни, а значением - словарь с информацией об артисте, дате выхода и количестве прослушиваний. В случае, если песни с одинаковыми названиями исполняют несколько артистов, то необходимо вывести информацию обо всех из них.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 14

Вы с друзья решили открыть продуктовый магазин. Друзья отвечают за рекламу и операционную деятельность. Спустя несколько месяцев работы вы поняли, что часть продуктов не распродается и ее необходимо списывать, что негативно влияет на общую прибыль. Ваша задача состоит в том, чтобы на основе накопленных данных написать функционал приложения для автоматизации учета продаж продуктов.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *products.txt,* который хранит в себе информацию о продукте, категории, цены за 1 кг, количество проданных товаров.

Столбцы: *Category, product, Date(в формате ДД.ММ.ГГГГ), Price per unit(вещественное число), Count(вещественное число)*

Разделитель *«|».*

Кодировка: "utf-8-sig"

Задача 1.

Формат ввода: стандартный ввод

Формат вывод: стандартный вывод

Вы собрали уже большое кол-во данных и хотите подвести итог полученной выручки за весь период. Для этого воспользуйтесь данными из таблицы *products.txt.* И создайте новый файл *products_new.csv* куда запишите два столбца первый это название категории(*Category)*, второй это количество единиц продукции в этой категории(*countProduct*). После выполнения этой задачи запросите у пользователя категорию и напишите в консоль количество товара по этой категории.

Формат вывода: "В категории <Category> находится <countProduct> единиц товара"

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортной работы и эффективной автоматизации учета продаж следует отсортировать по группам в обратном алфавитном порядке категории товаров в таблице *products.txt* и в первой категории вывести самый дешевый товар.

Формат вывода: "В категории: <Category> самый дешевый товар: <product> его цена за единицу товара составляет <Price per unit>"

В задаче нельзя использовать сторонние сортировки, для решения воспользуйтесь быстрой сортировкой.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Напишите программу, которая по заданному названию продукта выводит его количество, а также общую стоимость всех товаров в формате:

“На складе данного товара осталось <*Count*> единиц на общую стоимость - <*total*>”, если по заданной категории ничего не найдено, то вывести “Такого продукта нет на складе”

Если по заданному запросу найдено несколько товаров, то вывести товар с самой большой общей стоимостью.

Значение в столбце total рассчитывается путем умножения стоимости единицы продукции на количество проданных единиц.

Программа должна постоянно самостоятельно запрашивать название категории и остановиться только в случае ввода слова ‘плесень’.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вам нужно написать программу, которая считывает данные из файла и преобразует их в словарь, а затем выполняет некоторые аналитические задачи.

1. Программа должна считывать данные из файла products.txt и выводить информацию о категории, цене за 1 кг и количестве проданных товаров. Если продукты повторяются, то необходимо суммировать количество проданных товаров и вычислять среднюю стоимость.
Например:

```
{
   "Apples": {
      "Category": "Fruits",
      "Price per kg": 2.5,
      "Sold Quantity": 150
   },
   "Bananas": {
      "Category": "Fruits",
      "Price per kg": 1.2,
      "Sold Quantity": 250
   },
```

```
  "Tomatoes": {
    "Category": "Vegetables",
    "Price per kg": 1.5,
    "Sold Quantity": 100
  }
}
```

2. Необходимо вывести список продуктов, продажи которых составляют менее 15% порога от общего объема продаж в каждой конкретной категории товаров.

Последним этапом необходимо записать полученный список в новый файл – products_limit.txt. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Программа должна считывать данные из файла products.txt и преобразовать их в хеш-таблицу, используя название продукта в качестве ключа и словарь с информацией о категории, цене за 1 кг и количестве проданных товаров в качестве значения. Вам необходимо рассчитать общую прибыль с продаж каждой категории продуктов.

Выведите список из 5 наиболее прибыльных категорий товаров.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 15

Вы встали перед выбором будущей профессии. Ваш учитель любезно предоставил вам данные по вакансиям в ведущих компаниях в формате csv файла. Вы поняли, что проанализировать данный файл вручную невозможно, после чего вам приходит идея написать функционал для более удобного понимания и представления информации.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *vacancy.txt,* который хранит в себе информацию о должностях, заработной плате, типе трудоустройства, компаниях, количестве сотрудников в компании.

Столбцы: *Salary(целое число), Work_Type, Company_Size, Role, Company.*

Разделитель *«;»*

.

Задача 1.

Вы поняли, что хотите сначала проанализировать какие максимальные зарплаты можно получить при разных должностях в разных профессиях. Для этого создайте таблицу vacancy_new.csv в которую запишите три столбца Work_Type, Role, Salary. При этом сама вакансия(Work_Type) должна иметь максимальный размер зарплаты в своей категории, если в категории несколько типов работы с одинковой максимальной ЗП, добавьте в файл Role идущую первой в алфавитном порядке и с самым большим кол-вом работников в компании. После этого выведите сумму и вакансию на которую может претендовать стажер(с маленькой буквы): <Role> - <зарплата>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для себя вы отметили, что хотели бы работать в компании с самой большой зарплатой, так как у вас много амбиций и желаний. Поэтому Вам необходимо отсортировать данные с помощью сортировки слиянием(в задаче нельзя использовать встроенные функции сортировок) по столбцу зарплата в компании в порядке убывания. После этого выведите в консоль информацию о компании с наивысшей зарплатой.

Формат вывода: “В компании <Company> есть профессия: <Role>, з/п в такой компании составит: <Salary>”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у Вас название профессии(Role), а на выход будет выдавать список вакансий, если ничего не найдено будет выводить: “К сожалению, ничего не удалось найти”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “устал”.

Поиск необходимо осуществить с помощью двоичного алгоритма поиска**.**

Формат ответа на запрос пользователя: “В <компании> найдена искомая вакансия: <вакансия>. З/п составит: <Salary>”

Поиск необходимо осуществлять в файле vacancy.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Программа должна считывать данные из файла vacancy.txt и хранить их в виде многомерного списка. Каждый словарь будет содержать данные о должности, заработной плате, типе трудоустройства, компании.

Реализуйте методы/функции, которые будут определять среднюю заработную плату (после вычета налога) для определенной компании. Процентная ставка налога зависит от размера заработной платы и составляет:

Заработная плата до 45000 – налог 8%

Заработная плата 45001 - 50000 – налог 10%

Заработная плата свыше 50001 – налог 13%

Формат списка:

Название компании: средняя заработная плата (после вычета налога)

Полученный список запишите в новый файл – vacancy_average.txt. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Программа должна считывать данные из файла vacancy.txt и хранить их в хеш-таблице. Ключом будет должность, а значением - данные о заработной плате, типе трудоустройства, компании и количестве сотрудников в компании. В случае, если ключ встречается несколько раз, необходимо разрешить коллизии посредством метода цепочек.

Выведите информацию о 3 должностях с наибольшей и 3 должностях с наименьшей зарплатами. Если встречается несколько должностей с одинаковой заработной платой, то необходимо вывести все должности.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 16

Образованному человеку достаточно прочитать десять книг, но чтобы узнать, что это за книги необходимо прочитать несколько тысяч. Но мы живем в эпоху цифровой трансформации, в которую можно найти огромное количество информации. У вас есть csv файл с данными о книгах и их рейтинге, хранящихся в школьной библиотеке. Вам необходимо написать функционал для автоматизации поиска и подбора книг для чтения.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *books.txt,* который хранит в себе информацию о ISBN, авторе, дате выпуска, названии, рейтинге.

Столбцы: *book_id, isbn, authors, original_publication_year, original_title, ratings_1(вещественное число).*

Разделитель *«%».*

Задача 1.

Вы решили, что читать книги с плохим рейтингом не стоит, но некоторым авторам с большим рейтингом можно дать шанс. Поэтому Вы решили создать отдельный файл *books_new.csv* в котором создадите всего два столбца: *authors, ratings_authors,* где рейтинг автора рассчитывается из среднего значения всех оценок его книг(округлите до двух знаков после запятой). После этого выведите автора с самым маленьким рейтингом, если таких несколько выведите первого по алфавиту.

Формат вывода: "*<authors> - <ratings_authors>*"

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Теперь когда вы владеете документом со всеми книгами и их рейтингом, Вам хочется попробовать прочитать три книги с наилучшим рейтингом, чтобы понять а что в них такого особенного. Для этого вам необходимо отсортировать документ по столбцу с рейтингом с помощью сортировки слиянием. После сортировки выведите три книги с лучшим рейтингом в формате:

"<original_title> - <authors> - <ratings_1>"

В задаче нельзя использовать встроенные функции сортировок.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с вашим файлом. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у вас имя автора, а на выход будет выдавать список книг и их рейтинг, если ничего не найдено будет выводить: “Данного автора в этой библиотеке нет”. Программа должна всегда запрашивать название. Прекратить свою работу она сможет только после ввода “автор”.

Поиск необходимо осуществить с помощью двоичного алгоритма поиска**.**

Формат ответа на запрос пользователя:

“<название книги> - <isbn> - <book_id> - <ratings_1>”

Поиск необходимо осуществлять в файле books.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Прочитайте данные из файла books.txt и создайте многомерный список. Каждый словарь будет содержать информацию об одной книге: автор, название, дата выпуска, рейтинг и ISBN.

Реализуйте методы/функции, с помощью которых вы сможете вывести топ 10 авторов с наибольшим количеством книг. Если у книги несколько авторов, то ее необходимо засчитывать каждому из них.

Формат вывода:

Александр Пушкин

Лев Толстой

Марк Твен

Полученный список запишите в новый файл – top_10_authors.csv. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Считайте данные из файла books.txt и создайте хеш-таблицу для быстрого поиска книг, где ключом будет ISBN, а значением - словарь с остальными данными книги (автор, название, дата выпуска, рейтинг). Реализуйте методы/функции, результатом работы которых будет список книг, которые были изданы в 2000-2007 годах и имеют наивысший рейтинг (необходимо определить, книгу с наивысшим рейтингом в каждом году).

Формат вывода:

Год: 2005

Стивени Майер Сумерки 4,54

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 17

В компьютерной игре возникла новая зона с опасными монстрами, которые мешают героям продвигаться. Вы попали в команду разработчиков игры, которые решили создать программу, которая помогла бы героям улучшить свои навыки и повысить их шансы на победу над монстрами. В их команде помимо вас есть эксперты по игровой механике, балансировке и программированию, чтобы разработать такую программу.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *monster_game.txt,* который хранит в себе следующую информацию: Название монстра; Возможности(регенерация, дополнительный ход или усиление атаки); Вероятность возможности; Атака; Защита; Урон; Здоровье; Скорость;

Столбцы: *MonsterName, opportunity, probability, attack, protection, health, speed*

Разделитель *«$».*

Задача 1.

Ваш первый спринт заключается в том, что необходимо реализовать программу, которая подсчитает какая мощность силы у возможности монстра. Для этого создайте таблицу в файле monster_opportunity.csv со столбцами: название монстра(MonsterName), Название возможности(opportunity), мощность силы(power) – округлите до 3х знаков после запятой.

Мощность для регенерации высчитывается как процентная составляющая столбца здоровья (health * %). Значение процента берется из столбца - вероятность возможности

Мощность для дополнительного хода рассчитывается, как процент вероятности возможности от суммы всех сил (sum* %).

Мощность усиления атаки рассчитывается, как процент вероятности возможности от значения величины атаки (attack * %).

После создания таблицы выведите какая мощность силы получилась у первых пяти монстров.

В формате: число(значение мощности)

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Ваш тим-лид дал вам новую задачу и звучит она так: необходимо отсортировать возможности в алфавитном порядке с помощью быстрой сортировки (в задаче нельзя использовать встроенные функции сортировок!).

После этого продемонстрируйте результат вашей работы и выведите первых пять монстров в формате: “<MonsterName> имеет возможность: <opportunity>”

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей имя монстра а на выход будет выдавать его характеристику, если ничего не найдено будет выводить: “Ого, вам попался новый монстр! БЕГИТЕ!”. Программа должна всегда запрашивать название монстра. Прекратить свою работу она сможет только после ввода “мир”.

Формат вывода: «<MonsterName>: *<attack>, <protection>, <health>, <speed>*»

Поиск необходимо осуществлять в файле monster_game.txt

Поиск необходимо осуществить с помощью алгоритма с асимптотической сложностью O(n).

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Необходимо сделать список с опытом, который можно получить, если победить монстров. Для расчета опыта необходимо использовать следующий алгоритм. Необходимо сложить все значения полей атака, защита, здоровье, скорость, и в зависимости от значения поля возможности умножить значение соответствующего поля на 1,5. Т.е. если значение поля возможность «усиление атаки», то умножать на 1,5 значение поля «атака», если значение поля возможность «регенерация», то умножать на 1,5 значение поля «здоровье», если значение поля возможность «дополнительный ход», то умножать на 1,5 значение поля «скорость». Например,

*«Смертоносная суккуба, усиление атаки, 90, 8, 6, 2, 5» -> «Смертоносная суккуба, 25» // 8*1,5+6+2+5*

*«Кровожадный эльф, регенерация, 89, 0, 9, 8, 3» -> «Кровожадный эльф, 24» // 0+9+8*1,5+3*

Сформируйте список с именем монстра и значением опыта, выведите первые 10 элементов на экран.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Команда хочет понять, какой из наборов монстров является самым сильным по полю «атака» в перечне. Сформируйте хэш-таблицу, в которой в качестве ключа будет использоваться имя монстра, а в качестве значения сумма значений поля «атака» для этого имени монстра.

Выведите первые 10 элементов на экран.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 18

Ваш учитель по информатике предложил вам выбрать язык программирования, который вы будете изучать в этом году. Он выдал вашему классу csv файл с описанием языков программирования. Реализуйте функционал, который позволит не ошибиться с выбором языка программирования для изучения.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

```
if imag == 0.0 and real == 0.0: return complex_zero
...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *languages.csv,* который хранит в себе информацию о названии, типт, дате создания, создателе, кол-ве книг в библиотеке.

Столбцы: *title, type, appeared, creators, book_count.*

Разделитель *«;»*.

Задача 1.

Первым делом Вы решили определить в каких категориях больше всего есть книг для обучения, чтобы быть максимально уверенным в том, что вам хватит литературы и она сможет ответить на все ваши вопросы. Поэтому создайте новый файл с названием count_book.*txt,* формат каждой строки файла должен быть: “Книг для изучения <type> в библиотеке нашлось: <count>”. Также выведите количество книг в библиотеке по языкам запросов. Ответ выведите в формате: “Для изучения этой темы можно воспользоваться <count> книгами”.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом преподавателя вы решили его отсортировать по типу языков в алфавитном порядке. Для этого воспользуйтесь сортировкой, которая имеет среднюю сложность O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые три строчки в формате:

<type 1>:
<title 1> - создатель <creators>
…
<title M> - создатель <creators>
<type N>
<title 1> - создатель <creators>
…
<title M> - создатель <creators>

Где $N < M \leq 3$, то есть общее количество выводимых языков программирования равно трем.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей название языка программирования, а на выход будет выдавать его создателя и год создания, если ничего не найдено будет выводить: “Хм.. Вы уверены, что слышали об этом ЯП?”. Программа должна всегда запрашивать название ЯП. Прекратить свою работу она сможет только после ввода “0”.

Формат вывода ответа: <title> был создан: <creators> в <appeared>

Поиск необходимо осуществить с помощью двоичного поиска в файле languages.csv.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вы решили сделать свой рейтинг языков программирования из набора на основе данных о дате создания языка и количестве книг. Рейтинг языка будет рассчитываться по следующей формуле:

R=(book_count)/(2023-appeared)

Сформируйте список из названия языка и его рейтинга, выведите его содержимое на экран.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Для более быстрого поиска по названию языка программирования, создайте хэш-таблицу, где ключом будет являться название языка программирования, а значением строка с информацией о нем (тип, дата создания, создатель, количество книг).

Выведите полученную хэш-таблицу на экран.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 19

Астронавты, оказавшиеся на космической станции, ощутили, что время замерло в безмолвии космоса. Часы, которые обычно ярко сверкали и отсчитывали бесконечные минуты, теперь погрузились во мрак и несомненно потеряли свою силу. Они стали бесформенными и безжизненными, словно холодные камни, лишенные всякого движения. Астронавты, ошеломленные этим необычайным событием, осознали, что им придется восстановить работу этих часов, чтобы вернуться к нормальной жизни и ориентироваться во времени. Вы один из астронавтов, оказавшихся на космической станции. Примите участие и помогите всем остальным наладить ход времени.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
```

real – описание аргумента

imag – описание аргумента

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *astronaut_time.txt,* который хранит в себе информацию о номере часов; номере станции; номере каюты астронавта; времени остановки часов(ЧЧ:ММ:СС); кол-во часов простоя(целое число).

Столбцы: *WatchNumber, numberStation, cabinNumber, timeStop, count*

Разделитель *«>».*

Задача 1.

Восстановите время в каждой каюте. Для этого создайте новый файл формата csv с названием new_time.csv, куда добавьте такие столбцы: WatchNumber, numberStation, cabinNumber, timeStop, timeNow.

Где значения из столбца рассчитываются по формуле: timeNow = timeStop + count.

Обратите внимание, что в часах 24 часа, 60 минут и 60 секунд.

После этого выведите действительное время для своей каюты номер 98-OYE в формате:

<timeNow> - действительное время для каюты: <cabinNumber>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по номерам станций. Для этого воспользуйтесь сортировкой с вычислительной сложностью алгоритма — O(n2). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые три станции, которым нужна помощь в формате:

На станции <numberStation> в каюте <cabinNumber> восстановлено время. Актуальное время: <timeNow>

Где актуальное время рассчитывается по формуле: timeNow = timeStop + count.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей номер каюты, а на выход будет выдавать время остановки и восстановленное время, если ничего не найдено будет выводить: “В этой каюте все хорошо”. Программа должна всегда запрашивать номер каюты. Прекратить свою работу она сможет только после ввода “none”.

Формат вывода ответа:

В каюте <cabinNumber> восстановлено время (время остановки: <timeStop>). Актуальное время: <timeNow>

Где timeNow = timeStop + count.

Поиск необходимо осуществить с помощью алгоритма с асимптотической сложностью O(n) в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вы хотите понять в какое время чаще останавливались часы в каютах. Для этого сформируйте два списка, в одном из которых будет информация о станциях, где время остановки в промежутке от 00.00 до 12.00, а во втором от 12.01 до 23.59.

На основании сформированных списков выведите следующую информацию:

<длина списка 1> станций остановилось с период с 00.00 до 12.00.

<длина списка 2> станций остановилось с период с 12.01 до 23.59.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Чтобы поиск по каютам и станциям был эффективнее, создайте хэш-таблицу, в которой в качестве ключа будет использоваться номер каюты, а в качестве значения вся остальная информация о каюте.

Выведите информацию о первых 10 каютах в сформированной таблице.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 20

Ваши родители решили приобрести новый автомобиль, но не знают с чего начать подбор. В интернете вы нашли файл с описанием автомобилей, имеющихся в наличии. Ваша задача состоит в том, чтобы реализовать функционал, помогающий определиться с выбором.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл cars.csv, который хранит в себе информацию о марке, модели, пробеге, цене, дате и цвете.

Столбцы: price, year, manufacturer, model, odometer, paint_color

Разделитель «;».

Кодировка: “utf-8”

Задача 1.

Ваши родители определили для себя, что точно хотят машину красного цвета. Составьте для них файл red_car.txt со всеми машинами красного цвета, где каждая строчка файла будет в формате: У <manufacturer> <model> есть машина красного цвета. Ее стоимость <price>.

После этого ваши родители захотели посмотреть машины марки volvo, красного цвета. Выведите в консоль запись о машинах volvo красного цвета в формате:

<manufacturer>

<model 1> есть машина красного цвета. Ее стоимость <price>

…

<model N> есть машина красного цвета. Ее стоимость <price>

где N - кол-во машин, которые подходят под характеристики.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать машины по увеличению их пробега. Для этого воспользуйтесь сортировкой вставками. В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые три машины, которые имеют самый маленький пробег в формате:

Вам могут подойти:

<manufacturer> <model>; Цвет: <paint_color> ; Пробег: <odometer>; Цена: <price>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей марку машины и ее цвет(через пробел), а на выход будет выдавать все подходящие модели с их характеристиками, если ничего не найдено будет выводить: “По данным параметрам ничего не найдено”. Программа должна всегда запрашивать данные о машине. Прекратить свою работу она сможет только после ввода “car none”.

Формат вывода ответа:

По вашему запросу: <manufacturer> <paint_color> найдены следующие варианты:

1. <manufacturer> <model> цена <price>, пробег данной машины составляет <odometer>

…

N. <manufacturer> <model> цена <price>, пробег данной машины составляет <odometer>

где N - кол-во записей подходящих под условие

Поиск необходимо осуществить с помощью последовательного алгоритма поиска. в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Компания по продажам автомобилей прислала вам уведомление, что в настоящий момент предусмотрена скидка 12% на автомобили с пробегом более 70000 км. Сформируйте список с новыми ценами на автомобили. В списке должны содержаться марка автомобиля, модель автомобиля, пробег, новая цена.

Выведите информацию о первых 10 автомобилях в сформированном списке.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вам необходимо посчитать, сколько моделей каждой марки представлено в наборе данных. Составьте хэш-таблицу, где ключом будет название марки автомобиля, а значением – количество автомобилей этой марки.

Выведите полученную информацию согласно хэш-таблице для марок: gmc, chevrolet, toyota, ford, jeep, nissan, ram, cadillac, honda, dodge.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 21

Сегодня вам предстоит побывать в качестве героя волшебной сказки. У ваших друзей и вас собралось огромное количество рецептов для волшебных зелий, но в аптечке у сказочных героев закончились травы для их волшебных зелий. Ваша задача помочь им написать программу, которая поможет определить какие травы следует собрать чтобы восстановить запасы и помочь всем жителям сказочного мира.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл magical.txt, который хранит в себе информацию о названии зелья; кол-ве, которое есть в аптеках; названии травы_1; названии траввы_2; названии травы_3

Столбцы: magicaPotions,count,magic_herbs_1,magic_herbs_2,magic_herbs_3

Разделитель «№».

Кодировка: “utf-8”

Задача 1.

В переданном вам файле, есть зелья которых категорически не хватает. Жители такие зелья пометили количеством равным -1. Чтобы срочно собрать необходимы травы нужно создать файл magicaPotions.csv в который запишите два столбца: название травы(magic_herbs), количество травы(count_herbs). Подсчитайте общее количество травы, которое необходимо собрать и выведите в консоль в формате: Общее кол-во всех трав <count>. Подробнее смотрите в файле.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по названию зелий в алфавитном порядке. Для этого воспользуйтесь сортировкой с вычислительной сложностью алгоритма — O(n2). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых три зелья, которые имеют в своем составе иван-чай в формате:

Зелье <magicaPotions 1> имеет в свое составе иван-чай

…

Зелье <magicaPotions N> имеет в свое составе иван-чай

где N = 3

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей название травы(magic_herbs_1), которая используется первой, а на выход будет выдавать зелье с самым маленьким количеством в аптеках, если ничего не найдено будет выводить: “Такую траву мы не используем”. Программа должна всегда запрашивать данные о траве. Прекратить свою работу она сможет только после ввода “стоп”.

Формат вывода ответа:

По вашему запросу: <magic_herbs_1> найден следующий вариант:

<magic_herbs_1> используется в <magicaPotions>, его количество составляет : <count>

Поиск необходимо осуществить с помощью двоичного поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Вашему герою необходим перечень искристых зелий, которые есть в наличии, чтобы передать его феям. Сформируйте список таких зелий, с указанием их количества. В случае, если их нет в наличии в аптеке, включать их в список не нужно. При этом, если название зелья повторяется, считать их как разные зелья (т.к. состав отличается).

Выведите данный список на экран в формате:

<название зелья> осталось в наличии <количество> штук.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Проанализировав данные, вы поняли, что одно и то же зелье можно приготовить, используя несколько рецептов. Сформируйте хэш-таблицу, где ключом будет название зелья, а значением количество рецептов.

Выведите зелье с максимальным количеством рецептов в формате:

Зелье с максимальным количеством рецептов <название зелья> - <количество рецептов>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 22

Вы устроились программистом в компанию, которая занимается продажей компьютерной техники. На данный момент учет всех устройств ведется в Excel-таблице, что является сложным, так как объем продаж растет каждый месяц. Вам выгрузили данные в формате csv и попросили реализовать функционал, чтоб автоматизировать учет и подбор устройств.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

```
if imag == 0.0 and real == 0.0: return complex_zero
...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *devices.csv,* который хранит в себе информацию о компании, модели, типе, разрешении экрана, параметрах памяти и цене.

Столбцы: *Company, Product, TypeName, Inches, Cpu, Ram, Memory, Gpu, Price*

Разделитель *«;»*.

Задача 1.

Для того, чтобы вам было легче ориентироваться в количестве товаров Вам необходимо определить количество устройств в каждой компании. В исходном файле подсчитайте количество устройств от каждой компании и запишите в файл count_company.csv, где будет два столбца: Company, countProduct. После этого выведите количество товаров от компании Asus в формате:

От компании Asus у нас есть <countProduct> товаров.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по названию компании в обратном алфавитном порядке. Для этого воспользуйтесь сортировкой с средней сложностью O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять компаний в формате:

<Company> - <Product> - <Price>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей модель устройства, а на выход будет выдавать информацию по ней, если ничего не найдено будет выводить: “У нас нет данного устройства”. Программа должна всегда запрашивать данные о модели товара. Прекратить свою работу она сможет только после ввода “666”.

Формат вывода ответа:

По вашему запросу: <Product> найдены следующие варианты:

<Company> <Product> - тип устройства: <TypeName>; Разрешение экрана - <Inches>; Цена - <Price>

Поиск необходимо осуществить с помощью последовательного алгоритма поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Руководство компании хочет посчитать за реализацию продуктов какой из компаний можно получить максимальную прибыль. Сформируйте список, в котором будут содержаться названия компаний и сумма за продажу всей техники, имеющейся в базе.

Выведите получившуюся информацию на экран в формате:

Если продать все ноутбуки <Название компании> можно заработать <сумма цен>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Так как база данных устройств будет расширяться, необходимо сделать поиск по устройствам эффективнее. Сгенерируйте список устройств со значениями хэш-функции от полей Company+’ ’+Product.

Для хэширования необходимо использовать следующий алгоритм.

*hash(s)=s[0]+s[1]*p+s[2]*p^2+⋯+s[n-1]*p^(n-1) mod m=∑_(i=0)^(n-1)⁡〖s[i]*p^i mod m,〗*

где p и m - некоторые выбранные положительные числа, n – количество символов в строке.

Рекомендации по выбору чисел p и m.

Если входные данные состоят только из строчных букв английского алфавита, можно взять p = 31. Если же входные данные могут содержать как прописные, так и строчные буквы, то возможен выбор p = 53. Если используются прописные и строчные буквы русского алфавита, а также символ пробел, то возможет выбор p = 67.

значение числа m = 109+9.

Для вычисления хэша строки s, которая содержит только строчные буквы необходимо преобразовать каждый символ строки s в целое число. Можно использовать преобразование a →1, b →2, ... z → 26 . Преобразование a → 0 не является хорошей идеей, поскольку тогда хэши строк a, aa, aaa, ... все оцениваются как 0.

Выведите первые 10 значений сформированной таблицы.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 23

В волшебном королевстве, окутанном тайной и магией, произошло несчастье. Магическое зеркало, которое соткано из самых редких и мощных трав, сломалось. Это зеркало было не только прекрасным украшением в зале предсказаний, но и надежным источником информации о будущем для жителей королевства. Ваша задача: создать сис-му пользователей и найти того, кто сломал зеркало.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл history_mirror*.txt,* который хранит в себе информацию дате использования; ФИО; результататах использования.

Столбцы: date, username, verdict

Разделитель *«=»*.

Задача 1.

Первым делом вы решили найти пользователей у которых вместо предсказания появилась ошибка. Для этого в столбце verdict найдите пользователей со значением error и запишите их в файл mirror_error.txt в формате:

У пользователя <Фамилия И.О> <date> появилась ошибка

После этого выведите в консоль значение самого позднего сообщения об ошибке.

В формате:

Ошибка была зафиксирована: <date> у пользователя <Фамилия И.О>

Обратите внимание, что формат вывода ФИО отличает от формата в таблице.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по дате в порядке не возрастания. Для этого воспользуйтесь сортировкой вставками. В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять самых ранних использования:

< date> - <username> - <verdict>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей их фамилию, а на выход будет выдавать информацию, которую предсказало им зеркало, если ничего не найдено будет выводить: “Вы не использовали зеркало”. Программа должна всегда запрашивать данные о пользователе. Прекратить свою работу она сможет только после ввода “mirror”.

Формат вывода ответа:

Предсказание для <Фамилия И.О> - <verdict>

Поиск необходимо осуществить с помощью двоичного поиска поиска в исходном файле.

Обратите внимание, что данные в файле не отсортированы. Для сортировки можно воспользоваться встроенными сортировками или написать свою по столбцу username.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Королевство заметило, что последовательность запросов имеет значение для правильной работы зеркала. Зеркало работает лучше, когда каждому из запросов «Изменение характера» уменьшает силу зеркала, а запрос «Признание своей уникальности» восстанавливает ее на такую же величину. Соответственно, данные запросы должны быть сбалансированы.

Реализуйте проверку работы зеркала используя структуру данных «стек», определите каждый ли запрос «Изменение характера» нивелирован запросом «Признание своей уникальности».

Выведите ответ в формате:

«Запросы не изолированные» или «Запросы изолированные».

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Королевство хочет понять с какими запросами пользователи обращаются к зеркалу. Сформируйте хэш-таблицу, где ключом будет являться название запроса, а значением количество таких запросов.

Выведите 5 самых популярных запросов в формате:
<Название запроса> - <Количество таких запросов>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 24

В вашу компанию обратился крупный банк. Перед ними встала задача предложить клиентам максимально выгодный кэшбек по определенным категориям покупок. Они предоставили вам обезличенный набор данных в формате csv по совершенным покупкам за определенный период времени. Вам необходимо реалиовать функционал для получения данных в соответствии с запросами ниже.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

```
if imag == 0.0 and real == 0.0: return complex_zero
...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл *transactions.csv,* который хранит в себе информацию об идентификаторе пользователя, идентификаторе транзакции, дате, коде товара, количестве, описании товара, цене за 1 единицу.

Столбцы: *UserId, TransactionId, TransactionTime, ItemCode, ItemDescription, NumberOfItemsPurchased, CostPerItem*

Разделитель *«;»*.

Задача 1.

Для начала составьте список, в который добавьте товары только из категории упаковка(товары с этим словом в ItemDesription) и запишите в файл pack.txt в формате: <ItemDescription> - <CostPerItem>. После этого выведите самый дорогой товар в этой категории.

В формате:

Самый дорогой товар в категории упаковка: <ItemDescription>, цена такого товара составит: <CostPerItem>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по TransactionId в порядке не убывания. Для этого воспользуйтесь сортировкой с средней сложностью O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять товаров в формате:

<TransactionId> - <ItemDescription> - <CostPerItem>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей идентификатор транзакции, а на выход будет выдавать информацию по ней, если ничего не найдено будет выводить: “нет такой транзакции”. Программа должна всегда запрашивать данные о транзакции. Прекратить свою работу она сможет только после ввода “none”.

Формат вывода ответа:

По вашему запросу: <TransactionId> найден следующий вариант:

<UserId> <TransactionId> <TransactionTime> <ItemCode> <ItemDescription> <NumberOfItemsPurchased> <CostPerItem>

Поиск необходимо осуществить с помощью линейного алгоритма поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Перед вами стоит задача определить по имеющемуся набору данных, сколько покупок было совершено в определенный период времени.

Сформируйте два списка, в одном из которых будут храниться даты покупок в 2019 году, а во втором в 2018. Для каждого из списков посчитайте, сколько покупок было совершено с 07.00 до 12.00.

Выведите результат в формате:

<Длина списка покупок в 2018 году> покупок было совершено в 2018 году, из них <Количество покупок 07.00 до 12.00.>.

<Длина списка покупок в 2019 году> покупок было совершено в 2019 году, из них <Количество покупок 07.00 до 12.00.>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Для динамического подбора наиболее релевантного кэшбека необходимо получать аналитические выгрузки не реже чем раз в день. Наиболее эффективно сделать это при большом количестве транзакций встроенными инструментами языка программирования можно при использовании хэш-таблиц. Реализуйте хэш-таблицу, где ключом является идентификатор транзакции, а значением идентификатор пользователя.

Выведите первые 10 значений в сформированной таблице в формате

<Номер транзакции> <Номер пользователя>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 25

В компьютерной игре возникла новая зона с опасными монстрами, которые мешают героям продвигаться. Вы попали в команду разработчиков игры, которые решили создать программу, которая помогла бы героям улучшить свои навыки и повысить их шансы на победу над монстрами. В их команде помимо вас есть эксперты по игровой механике, балансировке и программированию, чтобы разработать такую программу.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *monster_game.csv,* который хранит в себе следующую информацию: Название монстра; Возможности(регенерация, дополнительный ход или усиление атаки); Вероятность возможности; Атака; Защита; Урон; Здоровье; Скорость;

Столбцы: *MonsterName, opportunity, probability, attack, protection, health, speed*

Разделитель *«,».*

Задача 1.

Ваш первая задача заключается в том, что необходимо реализовать программу, которая подсчитает какая максимальная мощность силы у возможности монстра. Для этого создайте таблицу в файле monster_opportunity.csv со столбцами: Название возможности(opportunity), максимальная мощность силы(power).

Мощность для регенерации высчитывается как процентная составляющая столбца здоровья(health * %). Значение процента берется из столбца - вероятность возможности

Мощность для дополнительного хода рассчитывается, как процент вероятности возможности от суммы всех сил (sum * %).

Мощность усиления атаки рассчитывается, как процент вероятности возможности от значения величины атаки (attack * %).

После создания таблицы выведите какая максимальная мощность силы получилась у возможности регенерации. В формате: Регенерация: <power>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Ваш тим-лид дал вам новую задачу и звучит она так: необходимо отсортировать возможности в обратном алфавитному порядку с помощью сортировки вставками(в задаче нельзя использовать встроенные функции сортировок!).

После этого продемонстрируйте результат вашей работы и выведите первых три монстра в формате: “<MonsterName> имеет возможность: <opportunity>, вероятность использования возможности равна <probability>” .

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей значение силы атаки, а на выход будет выдавать количество монстров, которых он сможет одолеть(те его сила атаки строго больше чем здоровье монстра и здоровье монстра не равно 0), если ничего не найдено программа должна вывести: “Вы очень слабы. Сходите и наберитесь опыта!”. Программа должна всегда запрашивать силу пользователя. Прекратить свою работу она сможет только после ввода “хватит”.

Формат вывода(при условии нахождения хотя бы одного монстра):

Вы сможете победить: N монстров

Где N – количество монстров со здоровьем меньше, чем уровень атаки пользователя

Поиск необходимо осуществлять в файле monster_game.csv

Поиск необходимо осуществить с помощью последовательного алгоритма поиска.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Ваша задача состоит в том, чтобы посчитать среднюю силу атаки для каждого класса монстров. Класс монстра определяется по второму слову в названии монстра, например

Класс «Шайтан»: Темный шайтан, Коварный шайтан, Злобный шайтан и т.д.

Класс «Колдун»: Хладнокровный колдун, Темный колдун, Могущественный колдун и т.д.

…

Сформируйте списки с каждым из классов монстров, и значением атаки.

Выведите результат в формате:

<Количество монстров класса 1> монстров класса <класс 1>, средняя сила атаки <среднее значение силы атаки по классу 1>

<Количество монстров класса 2> монстров класса <класс 2>, средняя сила атаки <среднее значение силы атаки по классу 2>

...

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Программисты хотят сделать рекомендательный модуль по выбору того или иного монстра, который находится в команде игрока для конкретного сражения. Так как время выбора героя играет важное значение для победы, важна скорость рекомендации. Для этого предлагается использовать хэш-таблицу где, в качестве ключа будет использоваться «название монстра»+«возможность»+ «вероятность возможности», а в качестве значения его возможность.

Выведите первые 10 значений сформированной таблицы в формате:

<Название монстра><возможность><вероятность возможности> - <возможность>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 26

Ваш учитель по информатике предложил вам выбрать язык программирования, который вы будете изучать в этом году. Он выдал вашему классу txt файл с описанием языков программирования. Реализуйте функционал, который позволит не ошибиться с выбором языка программирования для изучения.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
```

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *languages.txt,* который хранит в себе информацию о названии, типт, дате создания, создателе, кол-ве книг в библиотеке.

Столбцы: *title, type, appeared, creators, book_count.*

Разделитель *«$».*

Задача 1.

Первым делом Вы решили определить какие есть языки и книги **по типу: языки запросов**. Для этого создайте файл формата csv с названием program запишите в него такие столбцы: название языка программирования(title), информация о создателе(creators), а также количество книг по заданной тематике в библиотеке(*book_count).* Также выведите общее количество языков программирования по заданной тематике в формате: “Количество языков запросов равно: <count>”

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом преподавателя вы решили его отсортировать по названиям языков. Для этого воспользуйтесь сортировкой, которая имеет вычислительную сложностью алгоритма — O(n2). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые пять строчек в формате:

<title 1> – год создания <appeared> - <book_count>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей год создания ЯП, а на выход будет выдавать его название и создателя, если ничего не найдено будет выводить: “В этом году не было создано ЯП”. Программа должна всегда запрашивать год . Прекратить свою работу она сможет только после ввода “0000”.

Формат вывода ответа: В <appeared> был создан <title> его создатель: <creators>

Поиск необходимо осуществить с помощью линейного алгоритма поиска в файле languages.txt

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Ваша задача состоит в том, чтобы сформировать списки, содержащие перечень языков программирования по каждому типу с количеством книг по ним. По результатам формирования каждого из списков необходимо вывести следующую информацию:

<суммарное количество книг по типу 1> книг содержится в библиотеке по типу <тип 1>

<суммарное количество книг по типу 2> книг содержится в библиотеке по типу <тип 2>

…

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Для оптимизации поиска языка программирования по автору, создайте хэш-таблицу, где ключом будет являться автор, а значением название языка. В случае, если в таблице не указан создатель языка, в качестве ключа необходимо использовать конкатенацию строк «unknown»+«название языка».

Выведите информацию о первых 10 значениях в сформированной таблице в формате:

<Название автора> <Название языка>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 27

Астронавты, оказавшиеся на космической станции, ощутили, что время замерло в безмолвии космоса. Часы, которые обычно ярко сверкали и отсчитывали бесконечные минуты, теперь погрузились во мрак и несомненно потеряли свою силу. Они стали бесформенными и безжизненными, словно холодные камни, лишенные всякого движения. Астронавты, ошеломленные этим необычайным событием, осознали, что им придется восстановить работу этих часов, чтобы вернуться к нормальной жизни и ориентироваться во времени. Вы один из астронавтов, оказавшихся на космической станции. Примите участие и помогите всем остальным наладить ход времени.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
```

real – описание аргумента

imag – описание аргумента

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *astronaut_time.csv,* который хранит в себе информацию о номере часов; номере станции; номере каюты астронавта; времени остановки часов(ЧЧ:ММ:СС); кол-во часов простоя(целое число).

Столбцы: *WatchNumber, numberStation, cabinNumber(Двухзначное чило-трехзначный буквенный код), timeStop, count*

Разделитель *«,».*

Кодировка: «utf8»

Задача 1.

Восстановите время в каждой кающе. Для этого создайте новый файл формата txt с названием new_time.txt, Формат строк в файл должен быть оформлен в таком формате:

На станции <numberStation> в кающе <cabinNumber> восстановлено время. Актуальное время: <timeNow(ЧЧ:ММ:СС)>

Где актуальное время рассчитывается по формуле: timeNow = timeStop + count.

Обратите внимание, что в часах 24 часа, 60 минут и 60 секунд.

После этого выведите три первые записи из полученного файла.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать в алфавитном порядке по трехзначным буквенным шифрам номерам кают(столбец *cabinNumber*). Для этого

воспользуйтесь сортировкой вставками. В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые три станции, которым нужна помощь в формате:

На станции <numberStation> в каюте <cabinNumber> восстановлено время. Актуальное время: <timeNow(ЧЧ:ММ:СС)>

Где актуальное время рассчитывается по формуле: timeNow = timeStop + count.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей номер станции, а на выход будет выдавать время остановки и восстановленное время, если ничего не найдено будет выводить: “На этой станции все хорошо”. Программа должна всегда запрашивать номер станции. Прекратить свою работу она сможет только после ввода “stop”.

Формат вывода ответа:

На станции <numberStation> восстановлено время (время остановки: <timeStop>). Актуальное время: <timeNow(ЧЧ:ММ:СС)>

Где timeNow = timeStop + count.

Поиск необходимо осуществить с помощью последовательного алгоритма поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Прочитайте данные из файла astronaut_time.csv и сохраните их в списке для последующего анализа. Сгруппируйте данные по номеру станции и посчитайте общее время простоя часов на каждой из них. Выведите список из станций с временем простоя, равным 9.

Получившейся список запишите в новый файл – station_max_downtime.csv. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Прочитайте и загрузите данные из файла astronaut_time.csv в хеш-таблицу. Рассчитайте среднее количество часов простоя для каждой группы астронавтов. Определить астронавтов из одной группы можно по номеру станции (по числу, которое идет в начале). Например:

83 10-БЕМ 68-ЗОЧ 10:02:37 9

73 10-ЕЗА 78-GTZ 19:54:07 6

Выведите данные в следующем формате:

Номер группы астронавтов + Перечисление номеров кают + среднее время простоя.

Результат для примера, который представлен выше:

Номер группы: 10. Каюты: 68-ЗОЧ, 78-GTZ. Время простоя: 7,5

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 28

Ваши родители решили приобрести новый автомобиль, но не знают с чего начать подбор. В интернете вы нашли файл с описанием автомобилей, имеющихся в наличии. Ваша задача состоит в том, чтобы реализовать функционал, помогающий определиться с выбором.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
        ...
```

Теперь можете приступать к решению задач!

К задачам прикреплен файл cars.txt, который хранит в себе информацию о марке, модели, пробеге, цене, дате и цвете.

Столбцы: price, year, manufacturer, model, odometer, paint_color

Разделитель «$».

Кодировка: “utf-8”

Задача 1.

Ваши родители определили для себя, что точно хотят машину с пробегом меньше 10000. Составьте для них файл odometer_car.txt со всеми машинами подходящий под их запрос, где каждая строчка файла будет в формате:

<manufacturer> <model>; Цвет: <paint_color>; Пробег: <odometer>; Цена: <price>

После этого выведите родителям пять машин с пробегом меньше 10000 и цветом «серебро» в формате:

<manufacturer> <model> есть машина серебряного цвета. Ее стоимость <price> и пробег: <odometer>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать машины по возрастанию их цены. Для этого воспользуйтесь быстрой сортировкой. В задаче нельзя использовать встроенные сортировки! После сортировки выведите первые три самые дешевые машины машины в формате:

Вам могут подойти:

<manufacturer> <model>; Цвет: <paint_color> ; Пробег: <odometer>; Цена: <price>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей их бюджет(верхнюю и нижнюю границу включительно через пробел), а на выход будет выдавать все подходящие модели в рамках бюджета с их характеристиками, если ничего не найдено будет выводить: “К сожалению, под ваш бюджет ничего не удалось найти”. Программа должна всегда запрашивать данные о бюджете. Прекратить свою работу она сможет только после ввода “стоп”.

Формат вывода ответа:

Исходя из вашего бюджета: <min_input> - <max_input> найдены следующие варианты:

1. <manufacturer> <model> цена <price>, пробег данной машины составляет <odometer>

…

N. <manufacturer> <model> цена <price>, пробег данной машины составляет <odometer>

где N - кол-во записей подходящих под условие

Гарантируется, что первое вводимое число меньше второго.

Поиск необходимо осуществить с помощью двоичного поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Руководство компании заметило, что клиенты часто выбирают машины по цвету. Ваша задача состоит в том, чтобы сформировать списки машин каждого цвета. По каждому из формированных списков необходимо вывести следующую информацию:

<Количество машин цвета 1> машин цвета <название цвета 1> есть сегодня в наличии.

<Количество машин цвета 2> машин цвета <название цвета 2> есть сегодня в наличии.

...

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вам необходимо посчитать сколько моделей каждой марки представлено в наборе данных. Составьте хэш-таблицу, где ключом будет название марки автомобиля, а значением – количество автомобилей этой марки.

Выведите информацию из сформированной таблице в формате:

<название марки> <количество моделей>

для следующих марок: buick , chrysler, volvo, infiniti, lincoln, acura, hyundai, mercedes-benz, audi, bmw.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 29

Сегодня вам предстоит побывать в качестве героя волшебной сказки. У ваших друзей и вас собралось огромное количество рецептов для волшебных зелий, но в аптечке у сказочных героев закончились травы для их волшебных зелий. Ваша задача помочь им написать программу, которая поможет определить какие травы следует собрать чтобы восстановить запасы и помочь всем жителям сказочного мира.

Думаю, что Вы готовы приступать, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл magical.csv, который хранит в себе информацию о названии зелья; кол-ве, которое есть в аптеках; названии травы_1; названии траввы_2; названии травы_3

Номера травы отвечают за порядок приготовления.

Столбцы: magicaPotions,count,magic_herbs_1,magic_herbs_2,magic_herbs_3

Разделитель «,».

Кодировка: “utf-8”

Задача 1.

В переданном вам файле, есть несколько вариантов зельев. Подсчитайте общее количество для каждого из них, если количество равно -1, то замените его на ноль. Все категории запишите в файл magicaPotions.txt, где каждая строка будет соответствовать формату: <magicaPotions> в запасах еще есть - <count>

После этого выведите на экран количество остатков «Мощное Зелье» в формате:

Данного зелья осталось - <count>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по количеству зелий в порядке возрастания. Для этого воспользуйтесь сортировкой с средней сложностю O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять зелей, которые необходимо срочно восполнить:

Зелья <magicaPotions 1> осталось <count>

…

Зелья <magicaPotions N> осталось <count>

где N = 5

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей название травы(magic_herbs_3), которая используется последней в приготовлении зелья. На выход будет выдавать зелье с самым большим количеством в аптеках, если ничего не найдено будет выводить: “Такую траву мы не используем”. Программа должна всегда запрашивать данные о траве. Прекратить свою работу она сможет только после ввода “стоп”.

Формат вывода ответа(если нашлась такая трава):

По вашему запросу <magic_herbs_3> найден следующий вариант:

<magic_herbs_3> используется в <magicaPotions>, его количество составляет : <count>

Поиск необходимо осуществить с помощью линейного алгоритма поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Ваша задача состоит в том, чтобы посчитать количество зелий для каждого класса. Класс зелья определяется по второму (в некоторых названиях и третьему) слову в названии зелья, например

Класс «Лунный камень»: Зазеленевшее Лунный камень, Пузырящееся Лунный камень, Искристое Лунный камень и т.д.

Класс «Лед»: Переливающееся Лед, Искристое Лед, Летящее Лед и т.д.

...

Сформируйте списки с каждым из классов зелий, и количеством в наличии.

Выведите результат в формате:

<Количество зелий класса 1> зелий класса <класс 1>, общее количество зелий <сумма зелий в наличии по классу 1>

<Количество зелий класса 2> зелий класса <класс 2>, общее количество зелий <сумма зелий в наличии по классу 2>

...

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Сказочные герои хотят понять какой из ингредиентов является наиболее используемым для создания зелий. Чтобы это определить создайте хэш-таблицу, в которой в качестве ключа будет использоваться название ингредиента, а в качестве значения количество его упоминаний в рецептах по полям magic_herbs_1,magic_herbs_2,magic_herbs_3.

Выведите 5 наиболее популярных трав, используемых в рецептах в формате:

<Название травы> - <количество упоминаний>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 30

Вы устроились программистом в компанию, которая занимается продажей компьютерной техники. На данный момент учет всех устройств ведется в Excel-таблице, что является сложным, так как объем продаж растет каждый месяц. Вам выгрузили данные в формате csv и попросили реализовать функционал, чтоб автоматизировать учет и подбор устройств.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
   """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента

        """
        if imag == 0.0 and real == 0.0: return complex_zero
```

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *devices.txt,* который хранит в себе информацию о компании, модели, типе, разрешении экрана, параметрах памяти и цене.

Столбцы: *Company, Product, TypeName, Inches, Cpu, Ram, Memory, Gpu, Price*

Разделитель *«*»*.

Задача 1.

Вам поручили задание подсчитать количество ноутбуков(Ultrabook, Notebook, Netbook) с различными характеристиками Ram. В исходном файле подсчитайте количество устройств и запишите в файл count_company.txt, в формате:

Utrabook

<Ram 1> - <count>

…

<Ram N> - <count>

…

Notebook

<Ram 1> - <count>

…

<Ram N> - <count>

…

Netbook

<Ram 1> - <count>

…

<Ram N> - <count>

…

Где N - количество различных вариантов в этой категории. После этого выведите характеристику для Utrabook в формате:

Utrabook

<Ram 1> - <count>

…

<Ram N> - <count>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по названию компании в обратном алфавитном порядке. Для этого воспользуйтесь сортировкой с средней сложностью O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять компаний в формате:

<Company> - <Product> - <Price>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей название компании, а на выход будет выдавать информацию по самому дорогому устройству, если ничего не найдено будет выводить: “У нас нет данного устройства”. Программа должна всегда запрашивать данные о модели товара. Прекратить свою работу она сможет только после ввода “=”.

Формат вывода ответа:

По вашему запросу: <Company> найдены следующие варианты:

<Company> <Product> - тип устройства: <TypeName>; Разрешение экрана - <Inches>; Цена - <Price>

Поиск необходимо осуществить с помощью двоичного поиска в исходном файле.

Обратите внимание, что данные в файле не отсортированы. Для сортировки можете воспользоваться встроенными сортировками или из задачи 2.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Руководство компании хочет посчитать за реализацию продуктов, какой из компаний можно получить максимальную прибыль. Сформируйте список, в котором будут содержаться названия компаний и сумма за продажу всей техники, имеющейся в базе.

Выведите получившуюся информацию на экран в формате:

Если продать все ноутбуки <Название компании> можно заработать <сумма цен>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Вы разрабатываете интеллектуального помощника для менеджеров из отдела по холодным продажам. У менеджера есть скрипты, но количество техники достаточно большое, поэтому поиск работает неэффективно. Для быстрой выдачи подсказок реализуйте хэш-таблицу, где ключом является «компания»+ «продукт», а значением цена.

Выведите первые 10 значений сформированной таблицы в формате:

<Компания> <Продукт> <цена>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 31

В волшебном королевстве, окутанном тайной и магией, произошло несчастье. Магическое зеркало, которое соткано из самых редких и мощных трав, сломалось. Это зеркало было не только прекрасным украшением в зале предсказаний, но и надежным источником информации о будущем для жителей королевства. Ваша задача: создать сис-му пользователей и найти того кто сломал зеркало.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
 * Это описание функции foo
 *
 * @param str это описание аргумента str
 * @param pattern это описание аргумента pattern
 * @return это описание того, что вернет функция
 */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
```

"""

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл history_mirror*.csv,* который хранит в себе информацию дате использования; ФИО; результататах использования.

Столбцы: date, username, verdict

Разделитель *«,».*

Кодировка: «utf-8»

Задача 1.

До вас дошел слух, что человек с предсказанием Победа над смертью сломал зеркало. Для этого в столбце verdict найдите пользователей со заданным значением и запишите их в файл mirror_error.csv в столбцы: date, username

После этого выведите в консоль значение самого раннего сообщения о победе над смертью.

В формате:

Сообщение было зафиксировано: <date> у пользователя <Фамилия И.О.>

Обратите внимание, что формат вывода ФИО отличает от формата в таблице.

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать в алфавитном порядке по столбцу с вердиктом. Для этого воспользуйтесь сортировкой с вычислительной сложностью алгоритма — O(n2). В задаче нельзя использовать встроенные сортировки! После сортировки выведите полученные результаты в количестве четыре в формате:

< date> - <username> - <verdict>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей их Имя и Отчество(ввод осуществляется через пробел), а на выход будет выдавать информацию, которую предсказало им зеркало, если ничего не найдено будет выводить: “Вас не нашлось в записях”. Программа должна всегда запрашивать данные о пользователе. Прекратить свою работу она сможет только после ввода “stop”.

Формат вывода ответа:

Предсказание для <Фамилия И.О.> - <verdict>

Поиск необходимо осуществить с помощью алгоритма с асимптотической сложностью O(n) в исходном файле.

Если в файле нашлось несколько людей, то вывести их всех.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Жители королевства хотят понять, как часто используется зеркало. Помогите им получить эту информацию, сформировав отдельные списки для каждого года использования зеркала.

Результат обработки (для каждого из списков) выведите в формате:

В <год> году зеркало было использовано <количество записей в соответствующем списке>.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Необходимо сделать поиск по ФИО эффективнее. Сгенерируйте список пользователей со значениями хэш-функции от поля ФИО.

Для хэширования необходимо использовать следующий алгоритм.

$$hash(s)=s[0]+s[1]*p+s[2]*p^2+\cdots+s[n-1]*p^{(n-1)} \bmod m=\sum_{i=0}^{n-1} [s[i]*p^i \bmod m,]$$

где p и m - некоторые выбранные положительные числа.

Рекомендации по выбору чисел p и m.

Если входные данные состоят только из строчных букв английского алфавита, можно взять p = 31. Если же входные данные могут содержать как прописные, так и строчные буквы, то возможен выбор p = 53. Если используются прописные и строчные буквы русского алфавита, а также символ пробел, то возможет выбор p = 67.

Значение числа m = 109+9.

Для вычисления хэша строки s, которая содержит только строчные буквы необходимо преобразовать каждый символ строки s в целое число. Можно использовать преобразование a →1, b →2, … z → 26 . Преобразование a → 0 не является хорошей идеей, поскольку тогда хэши строк a, aa, aaa, … все оцениваются как 0.

Результат хэширования запишите в поле ID в файл users_with_hash.csv со следующей структурой:

ID, date, username, verdict

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект

Вариант 32

В вашу компанию обратился крупный банк. Перед ними встала задача предложить клиентам максимально выгодный кэшбек по определенным категориям покупок. Они предоставили вам обезличенный набор данных в формате txt по совершенным покупкам за определенный период времени. Вам необходимо реалиовать функционал для получения данных в соответствии с запросами ниже.

Вы готовы приступать к задачам, но сначала необходимо **создать репозиторий** для проекта и задач, которые Вы будете выполнять. Обязательно сделайте его **PUBLIC**, а то мы не сможем проверить решение вашей последней задачи. Каждую задачу вам необходимо будет правильно оформить и залить в репозиторий. *Код, который Вы напишете необходимо задокументировать, чтобы другие программисты могли понять, что делает код и за что отвечает. Примеры документирования приведены ниже.*

Пример документирования кода на языке C++

```
/**
  * Это описание функции foo
  *
  * @param str это описание аргумента str
  * @param pattern это описание аргумента pattern
  * @return это описание того, что вернет функция
  */
int foo(std::string str, std::string& pattern)
{
        ...
};
```

Пример документирования на языке Python

```
def complex(real=0.0, imag=0.0):
    """Описание функции complex.

        Описание аргументов:
        real – описание аргумента
        imag – описание аргумента
        """
```

if imag == 0.0 and real == 0.0: return complex_zero

...

Теперь можете приступать к решению задач!

К задачам прикреплен файл *transactions.txt* который хранит в себе информацию об идентификаторе пользователя, идентификаторе транзакции, дате, коде товара, количестве, описании товара, цене за 1 единицу.

Столбцы: *UserId, TransactionId, TransactionTime, ItemCode, ItemDescription, NumberOfItemsPurchased, CostPerItem*

Разделитель *«?»*.

Задача 1.

Для начала составьте список, в который добавьте товары только из категории набор(товары с этим словом в ItemDesription) и запишите в файл pack.csv в два столбца: ItemDescription и CostPerItem. После этого выведите самый дешевый товар в этой категории.

В формате:

Самый дешевый товар в категории набор: <ItemDescription>, цена такого товара составит: <CostPerItem>

В задаче запрещено использование сторонних библиотек(Pandas и др)

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка (для языка Python – PEP 257). После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 2

Для комфортного пользования файлом вы решили его отсортировать по ItemDescription в обратном алфавитном порядке. Для этого воспользуйтесь сортировкой с средней сложностью O(n log n). В задаче нельзя использовать встроенные сортировки! После сортировки выведите первых пять элементов в формате:

<TransactionId> - <ItemDescription> - <CostPerItem>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 3

Ввод: стандартный ввод

Вывод: стандартный вывод

Пришло время сделать наработку для интерфейса, который будет взаимодействовать с базой данных. Для этого Вам необходимо написать консольную программу, которая будет запрашивать у пользователей ItemCode, а на выход будет выдавать информацию по ней, если ничего не найдено будет выводить: “такого товара в базе нет”. Программа должна всегда запрашивать данные о ItemCode. Прекратить свою работу она сможет только после ввода “none”.

Формат вывода ответа(если найден хотя бы один элемент):

По вашему запросу: <ItemCode> найден следующий вариант:

<UserId> <TransactionId> <TransactionTime> <ItemCode> <ItemDescription> <NumberOfItemsPurchased> <CostPerItem>

Поиск необходимо осуществить с помощью двоичного поиска в исходном файле.

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория

Задача 4

Прочитайте и загрузите данные из файла transactions.txt в список. Проанализируйте данные, чтобы определить общую сумму покупок для каждого покупателя. Выведите 7 пользователей с наибольшей суммой покупок в формате:

<user id> <сумма покупок>

Полученный результат запишите в новый файл – best_buyer.csv. Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 5

Прочитайте и загрузите данные из файла transactions.txt в хеш-таблицу. Проанализируйте и выведите информацию о том, в какой месяц (вне зависимости от года) производится наибольшее количество покупок, а также среднюю стоимость товаров, купленных в этом месяце. В качестве ключа необходимо использовать номер месяца, а в качестве значения количество покупок и их общую стоимость. При помощи сформированной таблицы выведите информацию о покупках в следующем формате:

<Номер месяца> <количество покупок> <средняя стоимость покупок>

Не забудьте сделать комментарии к коду согласно стандартам документирования кода выбранного языка. После выполнения необходимо сделать локальные и удаленные изменения Вашего репозитория.

Задача 6

Ваш код будет использоваться программистом, которого возьмут на работу, поэтому он должен быть правильно оформлен и выложен на GitHub. Весь написанный код должен быть задокументирован согласно стандартам документирования кода выбранного языка.

Также необходимо оформить README.md для Вашего репозитория. Пункты, которые должны быть описаны:

1. Название проекта
2. Описание проекта
3. Оглавление (необязательно)
4. Как установить и запустить проект
5. Как использовать проект