Тестовое задание на frontend-стажировку в Контур

Задача: сделать консольную утилиту для фронтендера.

Утилита будет брать все файлы с расширением .js в текущей директории, находить в них все комментарии с TODO. Искать по ним, фильтровать, сортировать.
Мы уже написали за тебя метод для работы с консолью и методы для чтения из файлов. Осталось написать остальное =)

В исходниках несколько TODO-комментариев уже написано. Это сделано специально, чтобы тебе было на чем тестировать.

Для запуска используй команду '`node index.js`' При запуске специальный код ждет команды из консоли.

Сейчас он знает только команду `exit`, остальные нужно дописать тебе.

Все однострочные TODO-комментарии имеют одинаковое начало: два слеша, слово TODO (в любом регистре), пробел или двоеточие (или и двоеточие, и пробел; или пробел-двоеточие-пробел; или несколько пробелов) и дальше текст комментария.

Текст комментария в todo может быть представлен обычным текстом. Или же использовать специальную разметку:

```
// TODO {Имя автора}; {Дата комментария}; {Текст комментария}
```

После имени и даты обязательно ставится точка с запятой, а вот пробел между ними не обязателен.

Команды

Вот список команд, которые должно уметь обрабатывать приложение:

1. `exit` : завершение работы программы (уже реализовано)
2. `show` : показать все `todo`
3. `important` : показывать только `todo`, в которых есть восклицательный знак.
   В комментарии может присутствовать восклицательный знак (!), что означает, что это задача с высоким приоритетом.
4. `user {username}` : показывать только комментарии от указанного пользователя.
   Имя пользователя должно быть регистронезависимо. Пример команды: "`user veronika`". Сделай так, чтобы по начальным буквам имени пользователя список тоже показывался. То есть, результат команды "`user ve`" должен включать в себя такие же результаты, как и команда "`user veronika`" (и, может быть еще другие, если есть другие пользователи с именем, начинающимся на ve)

5. `sort {importance | user | date}` : выводит отсортированные todo
   Если аргумент `importance`, то сначала выводятся комментарии с восклицательными знаками, потом все остальные. Чем больше восклицательных знаков, тем выше приоритет и тем выше в списке этот комментарий.
   Если аргумент `user`, то выводятся задачи сгруппированные по пользователям, а в конце безымянные.
   Если аргумент `date`, то выводятся сначала самые новые, потом постарше, потом без дат.
   Примеры команд: "`sort importance`", "`sort user`", "`sort date`"
6. `date {yyyy[-mm-dd]}`: показывает все комментарии, созданные после переданной даты (включительно).
   Датой может быть только год, год с месяцем (через дефис) или год с месяцем и днем.
   Примеры команд: "`date 2015`", "`date 2016-02`", "`date 2018-03-02`".
   В ответ на команду "`date 2015`" ожидается список `todo`, которые были созданы в 2015 году и позже.

Если введена команда не из этого списка, то в консоль должен выводиться текст "`wrong command`".
В начале работы приложения, она должна писать в консоль приглашение ввести команду: "`Please, write your command!`".

Формат.

Выводи результаты в консоль в виде таблицы:
- каждая строка отображает один комментарий
- у таблицы должно быть пять колонок: важность, пользователь, дата, комментарий, имя файла (в котором найден этот todo).
- между ячейками должен быть разделитель — вертикальная черта (|). А от вертикальной черты до текста должно быть минимум два пробела отступа. Вот так:

```
!  |  pe  |  2018-03-02  |  sdkhsdfsdf  |  index.js
   |  pe  |  2018-03-02  |  sdkhsdfsdf  |  index.js
```

- если в комментарии есть восклицательные знаки, то в первой колонке нужно поставить символ !,
в остальных случаях ничего не ставить
- ширина колонок должна подбираться по самому длинному значению в колонке. Максимальная ширина колонок, не считая отступ до вертикальных черт: 1, 10, 10, 50, 15 соответственно. При необходимости обрезай значение, поставив в конце многоточие (...), но учти, что обрезанный текст вместе с многоточием должен влезть в максимальную

ширину колонки.
- у таблицы должен быть заголовок:

```
  !  |  user  |  date  |  comment  | fileName
```

- ширина клеток заголовка тоже должна подбираться по самому длинному значению в этом столбце
- от остальной таблицы заголовок отделяется строкой нужной длины со знаками минус (-) . Должно получиться как-то так:

```
  !  |  user  |  date  | comment  |  fileName
-----------------------------------------------
     |  pe    |  2012  | dddlsl   |  index.js
     |  pe    |  2012  | dddlsl   |  index.js
```

- добавь строку из минусов еще и в конце таблицы
- вывод комментариев в командах `show`, `important`, `user {username}`, `sort {type}` и `date {date}` должен отображаться в виде этой таблички.
- Если нет ни одного подходящего комментария, то выводится только заголовок таблицы и строка из минусов под ним. Вот так:

```
  !  |  user  |  date  | comment  |  fileName
-----------------------------------------------
```

Тесты.

К заданию приложены некоторые из тестов. Запускай их, чтобы проверить, что твое приложение ведет себя ожидаемо.
Чтобы запустить тесты:
- перейди в папку `tests`
- поставь все зависимости командой '`npm install`'
- запускай тесты командой '`npm test`'

Если у тебя MacOS или Linux, то тебе пригодится команда:
`$ chmod a+x ../index.js`

Формат готового задания.

После того, как приложение будет готово, упакуй все необходимые приложению файлы в

zip-архив с названием формата
Фамилия_Имя.zip. Тесты и, тем более, node_modules прикладывать к заданию не нужно.
Точкой входа должен быть файл index.js. Именно его будут запускать автотесты.
Предполагается, что тебе не понадобятся
никакие дополнительные пакеты, поэтому ставить зависимости автотесты не будут.
Проверь, что при запуске из корневой директории твоего проекта команды 'node index.js' все работает как ожидается.
Версия node.js, в которой будет запускаться приложение — 10.14
Версия npm — 6.4