

Школа бэкенд-разработки 2021 (осень)

🕒 7 сен 2021, 22:49:41

старт: 7 сен 2021, 19:54:21

финиш: 8 сен 2021, 01:54:21

до финиша: 03:04:29

длительность: 06:00:00

А. Состав заказа

Ограничение времени	2 секунды
Ограничение памяти	512.0 МБ
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Одной из важных частей онлайн-магазинов является обработка заказов. Заказ содержит в себе несколько товаров, каждый из которых приобретается в определенном количестве.

Но состав заказа не является строго фиксированным с момента оформления до момента фактической отправки заказчику. Пользователь может передумать покупать какой-либо товар (весь или частично) или, наоборот, решить увеличить количество какого-либо товара в заказе перед отправкой. В итоге может получиться даже так, что заказ в итоге нет смысла отправлять, так как в нём не осталось товаров.

Всё, что происходит с заказами в системе, фиксируется в формате событий "после изменения `event_id` в заказе `order_id` итоговое заказанное количество товара `item_id` равно `count`, а итоговое отмененное количество равно `return_count`". В итоге реальное количество товаров для отправки может быть вычислено как разница между `count` и `return_count`. Разница может получиться отрицательной - в рамках данной задачи это равносильно разнице, равной нулю (в причинах расхождения количеств в событии будет заниматься другой отдел).

Также у товара в заказе есть статус - `OK` или `CANCEL`. Это независимая от количества товара информация, обозначающая возможность отправки товара в данном заказе - если товар находится в статусе `CANCEL`, то его не надо отправлять, сколько бы штук товара не требовалось. Важно отметить, что статус `CANCEL` может сменить на `OK` в дальнейшем - к примеру, на складе появилась новая партия товара.

Вам требуется обработать список событий и вывести для каждого заказа итоговый список товаров для отправки. Для каждого товара необходимо учитывать **только последнее событие** (событие с максимальным значением `event_id`) среди событий, содержащих информацию о данном товаре. Если итоговый статус товара - `CANCEL`, то он не должен попасть в заказ. В заказ должны попасть только товары с ненулевым итоговым количеством для отправки. Выводить необходимо только заказы, в которых содержится хотя бы один товар для отправки.

Формат ввода

Входные данные представляют собой список событий в формате **JSON**.

Гарантии по формату JSON:

- нет запятых после последнего элемента массива;
- все имена полей и строки обернуты в двойные кавычки.

Гарантий по отступам и переводам строк **нет** - в примерах отступы и переводы строк используются для наглядности.

Обозначим количество событий в списке через N . Гарантируется, что $0 \leq N \leq 1000$.

Каждое событие содержит следующую информацию (порядок полей не является фиксированным):

- event_id**: идентификатор события.
- order_id**: идентификатор заказа.
- item_id**: идентификатор товара в заказе.
- count**: Итоговое количество единиц товара в заказе, запрошенное к отправке.
- return_count**: Итоговое количество единиц товара в заказе, отмененное к отправке.
- status**: статус товара в заказе.

Гарантируется, что идентификаторы и количества являются целыми числами ($0 \leq \text{event_id}, \text{order_id}, \text{item_id} \leq 2^{31} - 1$; $0 \leq \text{count}, \text{return_count} \leq 10^9$).

Гарантируется, что статус представлен только строками *OK* и *CANCEL*.

Гарантируется, что идентификаторы событий **уникальны**, но не гарантируется упорядоченность списка событий по возрастанию идентификаторов.

Формат вывода

Выведите в формате **JSON** список заказов для отправки.

Выводить JSON допустимо без дополнительных отступов и переводов строк - в примерах отступы и переводы строк используются для наглядности.

Имена полей необходимо выводить в **двойных кавычках**.

Допустимо выводить запятую после последнего поля объекта или последнего элемента массива.

Каждый заказ должен содержать следующую информацию:

- **id**: идентификатор заказа.
- **items**: список товаров в заказе.

Каждый товар в заказе должен содержать следующую информацию:

- **id**: идентификатор товара.
- **count**: количество товара для отправки.

Заказы и товары в рамках одного заказа можно выводить в любом порядке. Поля одной JSON-сущности так же можно выводить в любом порядке.

Ответ **не должен содержать**:

- товары с нулевым количеством для отправки.
- товары с итоговым статусом *CANCEL*.
- заказы с пустым списком товаров.

Пример 1

Ввод



```
[
  {
    "event_id": 1,
    "order_id": 1,
    "item_id": 1,
    "count": 1,
    "return_count": 0,
    "status": "OK"
  },
  {
    "event_id": 2,
    "order_id": 1,
    "item_id": 1,
    "count": 1,
    "return_count": 0,
    "status": "CANCEL"
  }
]
```

Вывод



```
[]
```

Пример 2

Ввод




Вывод



Ввод 

```
[
  {
    "event_id": 2,
    "order_id": 1,
    "item_id": 1,
    "count": 2,
    "return_count": 1,
    "status": "OK"
  },
  {
    "event_id": 1,
    "order_id": 1,
    "item_id": 1,
    "count": 2,
    "return_count": 0,
    "status": "OK"
  }
]
```


Вывод 

```
[
  {
    "id": 1,
    "items": [
      {
        "count": 1,
        "id": 1
      }
    ]
  }
]
```

Пример 3

Ввод 

```
[
  {
    "event_id": 2,
    "order_id": 2,
    "item_id": 1,
    "count": 3,
    "return_count": 1,
    "status": "OK"
  },
  {
    "event_id": 1,
    "order_id": 1,
    "item_id": 1,
    "count": 2,
    "return_count": 0,
    "status": "OK"
  },
  {
    "event_id": 3,
    "order_id": 3,
    "item_id": 1,
    "count": 2,
    "return_count": 2,
    "status": "OK"
  }
]
```

Вывод 

```
[
  {
    "id": 1,
    "items": [
      {
        "count": 2,
        "id": 1
      }
    ]
  },
  {
    "id": 2,
    "items": [
      {
        "count": 2,
        "id": 1
      }
    ]
  }
]
```

Примечания

При написании решения на Java можно выбрать компилятор "Java 8 + network + json-simple". В этом случае вы сможете воспользоваться библиотекой [json-simple](#) для парсинга и сериализации JSON.

Набрать здесь

Отправить файл

1

Отправить

Следующая