Вступительная работа в параллель ЛКШ.P.2025

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 недели
Ограничение по памяти:	256 мегабайт

Базовая версия вступительного задание представляет собой разработку консольного приложения для работы с API спортивной статистики, которое представлено нашим сервером.

На **HTTP** сервере хранятся следующие сущности: **Match**, **Team**, **Player**. Схема указана в замечании. Получить данные можно с помощью следующих endpoint'ов в формате **JSON**.

- `GET /matches` — список всех матчей
- `GET /teams` — список всех команд
- `GET /teams/{id}` — информация о команде по ID
- `GET /players/{id}` — информация об игроке по ID

Для выполнения задания необходимо:

- Получить **plain-token** (строка из 64 символов) у телеграм бота (`tg@lksh_p_2025_bot`).
- Этот токен необходимо передавать в заголовке `Authorization` при каждом запросе к API.

Функциональность

1. В начале в стандартный поток вывода нужно вывести список всех игроков, которые принадлежат к существующим командам. Игроков необходимо отсортировать в лексикографическом порядке их имени, каждого игрока один раз.
2. Обрабатывать запросы о статистике определенной команды

   ```
   > stats? "<имя команды>"
   ```

   На запрос вывести на новой строке три числа: количество побед команды, количество поражений, разницу забитых и пропущенных голов (<забитые голы> - <пропущенные голы>). Если такой команды не существует - вывести три нуля.
3. Обрабатывать запросы о количестве матчей, в которых соревовались два игрока.

   ```
   > versus? <player1_id> <player2_id>
   ```

   На запрос вывести на новой строке одно число: количество матчей, в которых первый игрок играл против второго игрока. Если такого игрока не существует - вывести нуль.

Формат входных данных

Каждый новый запрос указывается на отдельной строке. Запрос может быть *stats*? или *versus*?. После запроса *stats*? через пробел в кавычках указывается имя команды. После запроса *stats*? через пробел указывается два целых неотрицательных числа - идентификаторы первого и второго игрока.

Формат выходных данных

В начале в стандартный поток вывода нужно вывести список всех игроков, которые принадлежат к существующим командам. Игроков необходимо отсортировать в лексикографическом порядке их имени, каждого игрока один раз.

Ответ на каждый последующий запрос необходимо выводить на новой строке, разделяя числа пробелами.

Пример

стандартный ввод	стандартный вывод
stats? "Better" versus? 5 3 stats? "The Best" versus? 5 6	Andrew Karpenko Andrew Stankevich Danya Golov Egor Bugaev Khet Dzestelov Vanya Kabakov 1 1 -3 1 2 0 +5 0

Замечание

Рекомендуется протестировать своё приложение, тестовый сервер развернут по адресу `https://lksh-enter.ru`. Обратите внимание на используемый протокол.

Пример входных и выходных данных соответствует следующему содержимому сервера:

Пример ответа `GET /matches`:

```
[ {id: 1, team1: 10, team2: 20, team1_score: 5, team2_score: 1},
{id: 2, team1: 10, team2: 30, team1_score: 3, team2_score: 2},
{id: 3, team1: 20, team2: 30, team1_score: 4, team2_score: 3} ]
```

Пример ответа `GET /teams`:

```
[ {id: 10, name: 'The Best', players: [1, 2]},
{id: 20, name: 'Better', players: [3, 4]}
{id: 30, name: 'Worse', players: [5, 6]} ]
```

Пример ответа `GET /teams/{id}`:

```
{id: 10, name: 'The Best', players: [1, 2]}
```

Примеры ответа `GET /players/{id}`:

```
{id: 1, name: 'Danya', surname: 'Golov', number: 13}

{id: 2, name: 'Khet', surname: 'Dzestelov', number: 11}

{id: 3, name: 'Andrew', surname: 'Karpenko', number: 9}

{id: 4, name: 'Vanya', surname: 'Kabakov', number: 23}

{id: 5, name: 'Andrew', surname: 'Stankevich', number: 4}

{id: 6, name: 'Egor', surname: 'Bugaev', number: 99}
```

Примечание: Имена команд уникальны. Имена игроков и команд могут состоять из русских и английских букв, а так же содержать пробелы. Имя и фамилия игрока могут быть пустыми.