Задача 1

Напишите программу, которая будет фиксировать изменение. Для этого нужно составить словарь с ключами Аргентина (argentina) и Чили (chile). Значения по ключам: словари, в которых по каждому встреченному типу ландшафта записаны растения и животные, на них обитающие.

Формат ввода

Вводятся строки, в которых через # записаны: *A* (Аргентина) или *Ch* (Чили); ландшафт; растение (*plant*) или животное (*animal*); название.

Формат вывода

В файл **patagonia.json** запишите полученный словарь, списки в котором должны быть отсортированы сначала по типу – сначала животные, потом растения, внутри одного типа по алфавиту.

Пример

Ввод

```
A#glacier#animal#penguin
A#glacier#plant#tipchak
Ch#plateau#plant#bluegrass
A#glacier#animal#condor
A#plateau#animal#cougar
A#forest#animal#cougar
Ch#glacier#animal#ostrich
Ch#plateau#animal#condor
```

Вывод

```
{
    "argentina": {
        "glacier": [
            "condor",
            "penguin",
            "tipchak"
        ],
        "plateau": [
            "cougar"
        ],
        "forest": [
            "cougar"
        ]
    },
    "chile": {
        "plateau": [
            "condor",
            "bluegrass"
        ],
        "glacier": [
            "ostrich"
        ]
    }
}
```

Задача 2

Напишите программу, считающую количество бабочек определенного вида и определенного размера.

В файле **migration.json** записаны адрес и порт сервера, а также параметры для отбора бабочек:
serv – адрес сервера;
gate – порт;
kind – вид бабочек для поиска;
min_size – минимальный интересующий размер.

На сервере записан словарь с ключами – видами бабочек, значения по которым – списки словарей:
id;
size – размер;
place – место на яхте;
amount – количество.

Прочитав все данные с сервера, найдите всех бабочек указанного вида и с размером, не меньшим указанного. Запишите информацию о них в файл **butterflies.csv**, заголовки (разделители *):
место на яхте, размер, общее количество данного размера в данном месте
place, size, total

Строки в файле должны быть отсортированы по убыванию количества, затем по убыванию размера, затем по месту по алфавиту.
Проверять наличие искомого вида на сервере не нужно.

Пример (см. на следующей странице)

Ввод

```
# Содержимое файла migration.json
{
   "serv": "127.0.0.1",
   "gate": "5000",
   "kind": "monarch",
   "min_size": 7
}
# Данные на сервере
{
  "peacock's eye": [
    {
      "id": 42,
      "size": 8,
      "place": "wheelhouse",
      "amount": 12
    }
  ],
  "monarch": [
    {
      "id": 47,
      "size": 8,
      "place": "bank",
      "amount": 41
    },
    {
      "id": 12,
      "size": 8,
      "place": "mast",
      "amount": 35
    },
    {
      "id": 19,
      "size": 7,
      "place": "tackle",
      "amount": 41
    },
    {
      "id": 27,
      "size": 8,
      "place": "mast",
      "amount": 6
    },
    {
      "id": 5,
      "size": 5,
      "place": "stern",
      "amount": 21
    }
  ]
}
```

Вывод

```
place*size*total
bank*8*41
mast*8*41
tackle*7*41
```

Примечания

JSON файл из условия прилагается

Задача 3

Напишите программу, выбирающую в пампасах деревья с подходящей тенью и подходящего цвета.

Формат ввода

Через аргументы командной строки передаются:
хост и порт сервера;
произвольное число расцветок растительности.
Также могут быть переданы:
–*shadow* – минимальный размер тени, по умолчанию 0;
–*intensity* – характеристика цвета (*dark, light*...), по умолчанию пустая строка.

На сервере записана информация о растительном мире пампасов в виде словаря:
ключи – виды растительности;
значения – списки словарей:
name – название;
color – цвет;
drought – засухоустойчивость;
shadow – размер тени.

Формат вывода

Получив с сервера **все** данные, нужно выбрать из них те растения, для которых выполнены условия:
цвет без учета характеристики есть среди позиционных аргументов командной строки;
размер тени не меньше значения аргумента *shadow*;
в названии цвета присутствует характеристика *intensity*.

Отобранные данные нужно записать в файл **pampa.json** в виде списка списков:
[*название, вид, засухоустойчивость, тень*]
отсортированных по убыванию размера тени, затем убыванию засухоустойчивости, затем названию по алфавиту.

Пример

```
# Пример запуска
python3 solution.py 127.0.0.1 5000 green beige lilac --intensity dark
```

(см. на следующей странице)

Ввод

```
# Данные на сервере
{
  "tree": [
    {
      "name": "ombu",
      "color": "dark green",
      "drought": 50,
      "shadow": 150
    },
    {
      "name": "discoria",
      "color": "light green",
      "drought": 75,
      "shadow": 50
    },
    {
      "name": "white mosquito",
      "color": "lilac",
      "drought": 80,
      "shadow": 50
    }
  ],
  "grass": [
    {
      "name": "fescue",
      "color": "dark beige",
      "drought": 40,
      "shadow": 2
    },
    {
      "name": "laconose",
      "color": "dark golden",
      "drought": 40,
      "shadow": 2
    },
    {
      "name": "grasshopper",
      "color": "dark green",
      "drought": 40,
      "shadow": 2
    },
    {
      "name": "thistle",
      "color": "light beige",
      "drought": 30,
      "shadow": 1
    }
  ]
}
```

Вывод

```
[
    [
        "ombu",
        "tree",
        50,
        150
    ],
    [
        "fescue",
        "grass",
        40,
        2
    ],
    [
        "grasshopper",
        "grass",
        40,
        2
    ]
]
```