Перевернуть словарь

Напишите функцию, которая принимает словарь и возвращает новый словарь, в котором для каждого значения из исходного словаря содержится множество соответствующих ему ключей. В новом словаре ключи должны быть отсортированы по возрастанию.

```
>>> dict(invert({"a": 42, "b": 42, "c": 24}))
{24: {'c'}, 42: {'b', 'a'}}
```

Считаем символы в дзене Python

Выполните следующее задание:

1) Сохраните дзен Python (используя import this) в txt файл;
2) Выведите 10 самых популярных слов из полученного файла, обрабатывая файл построчно.

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Сложная сортировка

Напишите функцию, которая принимает на вход список студентов и сортирует их по ключу “длина имени + длина фамилии” в обратном порядке.

```
>>> students = [
...     {
...         'name': 'Artem',
...         'surname': 'Kravchuk',
...         'gpa': 2,
...     },
...     {
...         'name': 'Artem1',
...         'surname': 'Kravchuk',
...         'gpa': 2.1,
...     },
...     {
...         'name': 'a',
...         'surname': 'b',
...         'gpa': 5,
...     },
... ]
>>> custom_sort(students)
[{'name': 'Artem1', 'surname': 'Kravchuk', 'gpa': 2.1}, {'name': 'Artem', 'surname': 'Kravchuk', 'gpa': 2}, {'name': 'a', 'surname': 'b', 'gpa': 5}]
```

Правильная скобочная последовательность

Реализуйте функцию, которая по строке, состоящей из символов “)” и “(“ определяет, является ли она правильной скобочной последовательностью. Необходимо использовать deque.

```
>>> is_correct_brackets_seq('()')
True
>>> is_correct_brackets_seq('()()')
True
>>> is_correct_brackets_seq('(()')
False
>>> is_correct_brackets_seq('(())')
True
>>> is_correct_brackets_seq('(()(')
False
```