Metody programowania - Baca - Zadanie 1 - Sortowanie

Treść

Jesteś stażystą w swojej pierwszej pracy. Szef zlecił Ci głupie zadanie związane z przetwarzaniem danych pracowników firmy, a w dodatku ma nieuzasadnione wymagania. Chcesz zarobić na swój pierwszy samochód, więc nie zadajesz zbędnych pytań i je wykonujesz.

Posortuj w porządku rosnącym listę obiektów klasy Employee. O położeniu w pierwszej kolejności decyduje wartość pola 'person', w drugiej pola 'experience', a w trzeciej pola 'salary'. W przypadku pola 'person' o kolejności ma decydować pole 'surname', a dopiero kiedy nazwiska są takie same, to pole 'name'.

Sortowanie po każdym z tych pól ma być wykonywane osobno i za pomocą różnych algorytmów sortujących. Wykorzystaj (w wybranej kolejności) Cocktail Sort, Selection Sort i Count Sort w wersjach przedstawionych na wykładzie.

Fragment kodu

```
class Person:
    def __init__(self, name, surname):
        self.name = name
        self.surname = surname

    def __str__(self):
        return 'name: {:10} surname: {:15}'.format(self.name, self.surname)

class Employee:
    def __init__(self, person, experience, salary):
        self.person = person
        self.experience = experience
        self.salary = salary

    def __str__(self):
        return '{} experience: {:2} salary: {:5}'.format(self.person,
self.experience, self.salary)
```

Wejście

Na wejściu wprowadzane są cztery linie:

1. Linia z imionami (name)
2. Linia z nazwiskami (surname)
3. Linia z doświadczeniem (experience)
4. Linia z zarobkami (salary)

Każda linia zawiera elementy rozdzielone pojedynczymi spacjami i kończy się spacją. Należy je odczytać i w oparciu o nie wygenerować listę obiektów klasy Employee.

Wyjście

Na wyjściu w osobnych liniach mają wyświetlać się posortowane obiekty zgodnie z przedstawionymi powyżej metodami **str**, np.

```
name: Gertruda   surname: Haba           experience: 59 salary: 10140
```

Wymagania implementacyjne

1. Na Bacy dostępny jest Python w wersji 2.7. W praktyce oznacza to, że

- wejście powinno być wczytywane z użyciem funkcji raw_input zamiast input,
- nie wolno korzystać z ,,nowoczesnych" funkcjonalności (np. anotacje z typami danych).

2. Przedstawiony fragment kodu powinien zostać wykorzystany w niezmienionej formie w rozwiązaniu zadania.
3. Nie można korzystać z gotowych algorytmów sortujących - mają one być zaimplementowane od podstaw w osobnych funkcjach.
4. Nie można importować zewnętrznych bibliotek.

Przykładowe dane

Wejście:

```
Adam Bromir Bruno Abelard Adam Bromir Bruno Abelard Adam Bromir Bruno Abelard Adam
Bromir Bruno Abelard

Dachtera Dacka Dachtera Dacka Dachtera Dacka Dachtera Dacka Dachtera Dacka
Dachtera Dacka Dachtera Dacka Dachtera Dacka

8 25 56 2 24 59 19 38 8 25 56 2 24 59 19 38

7080 4500 6360 3120 8040 10140 7740 8880 8640 3540 3180 3240 7500 7020 9840 7800
```

Wyjście:

```
name: Adam       surname: Dachtera       experience:  8 salary:  7080
name: Adam       surname: Dachtera       experience:  8 salary:  8640
name: Adam       surname: Dachtera       experience: 24 salary:  7500
name: Adam       surname: Dachtera       experience: 24 salary:  8040
name: Bruno      surname: Dachtera       experience: 19 salary:  7740
name: Bruno      surname: Dachtera       experience: 19 salary:  9840
name: Bruno      surname: Dachtera       experience: 56 salary:  3180
name: Bruno      surname: Dachtera       experience: 56 salary:  6360
```

```
name: Abelard    surname: Dacka          experience:  2 salary:  3120
name: Abelard    surname: Dacka          experience:  2 salary:  3240
name: Abelard    surname: Dacka          experience: 38 salary:  7800
name: Abelard    surname: Dacka          experience: 38 salary:  8880
name: Bromir     surname: Dacka          experience: 25 salary:  3540
name: Bromir     surname: Dacka          experience: 25 salary:  4500
name: Bromir     surname: Dacka          experience: 59 salary:  7020
name: Bromir     surname: Dacka          experience: 59 salary: 10140
```