Lista 3 (Lab) Termin wysłania na SVN do 29.03.2020

Przy pisaniu programów (funkcji) **NALEŻY** stosować się do ogólnie przyjętego stylu programowania w języku Python. Proszę dokładnie przeczytać PEP 8 (tutorial, google python style) i PEP 257.

1. (5pt) Załóżmy, że reprezentujemy macierze kwadratowe w Pythonie następująco (dla rozmiaru macierzy n=3):

```
2.     ["1.1 2.2 3.3", "4.4 5.5 6.6", "7.7 8.8 9.9"]
```

   Napisz funkcję wykorzystując tylko listy składane, która dokonuje transpozycji takich macierzy (dowolnych rozmiarów) oraz zwraca wynik w tej samej postaci (można to zrobić w jednej linii kodu!).
3. (5pt) Napisz **generator** o nazwie "flatten", który przechodzi dowolną listę (również zagnieżdżoną) i podaje po kolei jej elementy: Na przykład dla listy

```
4.     l = [[1, 2, ["a", 4, "b", 5, 5, 5]], [4, 5, 6 ], 7, [[9, [123, [[123]]]],
   10]]
```

   Po wywołaniu: `print(list(flatten(l)))`, otrzymujemy:

```
       [1, 2, 'a', 4, 'b', 5, 5, 5, 4, 5, 6, 7, 9, 123, 123, 10]
```

5. (5pt) Wykorzystując, tylko listy składane (jako generatory) napisz program, który odczytuje plik tekstowy następnie pobiera ostatnią kolumnę, która zawiera informację o wielkości pliku, sumuje i wynik wyświetla na ekranie. Przykład pliku:

```
6.
7.   127.0.0.1 -  - "GET /ply/  HTTP/1.1" 200 7587
8.   127.0.0.1 -  - "GET /favicon.ico HTTP/1.1" 404 133
9.   127.0.0.1 -  - "GET /ply/bookplug.gif" 200 23903
10.     127.0.0.1 -  - "GET /ply/ply.html HTTP/1.1" 200 97238
11.     127.0.0.1 -  - "GET /ply/example.html HTTP/1.1" 200 2359
12.     127.0.0.1 -  - "GET /index.html" 200 4447
13.
```

   Dostajemy wynik:

```
       Całkowita liczba bajtów: 135667
```

14. (5pt) Rozważmy algorytm *QuickSort* napisany w języku Haskell:

```
15.
16.    qsort [] = []
17.    qsort (x:xs) = qsort elts_lt_x ++ [x] ++ qsort elts_greq_x
18.                   where
19.                     elts_lt_x = [y | y <- xs, y < x]
20.                     elts_greq_x = [y | y <- xs, y >= x]
```

21.

Napisz podobny program w języku Python wykorzystując

- składnie funkcjonalną (filter)
- operacje na listach składanych

22. (5pt) Poniżej w języku OCAML napisany jest program, który generuje wszystkie podzbiory

```
23.
24.
25.     let rec allsubsets s =
26.       match s with
27.         [] -> [[]]
28.       | (a::t) -> let res = allsubsets t in
29.                     map (fun b -> a::b) res @ res;;
30.
31.     # allsubsets [1;2;3];;
32.     - : int list list = [[1; 2; 3]; [1; 2]; [1; 3]; [1]; [2; 3]; [2];
   [3]; []]
33.
34.
```

Napisz podobny program w języku Python wykorzystując

- składnie funkcjonalną (map, lambda)
- operacje na listach składanych