Kurs języka Lua 2017

Lista zadań nr 5

Na zajęcia 3–4.04.2017

Za zadania z tej listy można uzyskać maksymalnie 6 punktów.
Styl kodu ma wpływ na ocenę jakości rozwiązania.

Zadanie 1. (3p) Napisz bezstanowy iterator `chain`, działający jak pythonowy itertools.chain. Iterator przyjmuje dowolną liczbę argumentów będących sekwencjami.

```
for x in chain({'a', 'b', 'c'}, {40, 50}, {}, {6, 7}) do
  print (x)
end
--> a \n b \n c \n 40 \n 50 \n 6 \n 7
```

Zadanie 2. (3p) Napisz bezstanowy iterator `zip`, działający podobnie jak pythonowy itertools.izip. Iterator przyjmuje dowolną liczbę argumentów będących sekwencjami i zwraca „listę argumentów" odpowiadających kolejnym pozycjom w podanych sekwencjach do momentu gdy którakolwiek z sekwenci się skończy.

```
for x, y in zip({'a', 'b', 'c', 'd'}, {40, 50, 60}) do
  print (x, y)
end
--> a 40 \n b 50 \n c 60
```

Zadanie 3. (3p) Napisz iterator `subsets`, który dla zadanego zbioru zwróci, w dowolnej kolejności, wszystkie jego niepuste podzbiory.
(Zamiast tworzyć nową tablicę dla każdego podzbioru, możesz wykorzystać jedną tablicę i wyłącznie modyfikować jej zawartość.)

```
for s in subsets{a=true, b=true, 3=true} do
  print (table.concat(s))
end
--> a \n b \n 3 \n ab \n a3 \n b3 \n ab3
```

Zadanie 4. (3p) Napisz dekorator `cache (f [, maxsize])`, działający w przybliżeniu jak pythonowy functools.lru_cache. Idea jest taka, żeby dla danej funkcji `f`, funkcja `cache` zwracała funkcję która działa jak `f`, ale spamiętuje wyniki dla pierwszych `maxsize` różnych zapytań (dla wartości `nil` spamiętuje wszystkie wyniki).
Zastanów się w jaki sposób efektywnie obsługiwać wywołania funkcji o dowolnej liczbie argumentów.
Uwaga. To zadanie będzie miało dalszy ciąg na kolejnej liście.