Kurs języka Lua 2017

Lista zadań nr 3

Na zajęcia 20–21.03.2017

Za zadania z tej listy można uzyskać maksymalnie 4 punkty.
W zadaniach 1–5 nie można korzystać z patternów.
Pamiętaj, że funkcję w Lua można zadeklarować na dwa równoważne sposoby:

```
function f(a, b) return a+b end
f = function(a, b) return a+b end
```

Zadanie 1. (1p) Do biblioteki `utf8` dopisz funkcję `reverse(s)` która poprawnie odwraca napisy w UTF-8.

```
utf8.reverse('Księżyc') --> 'cyżęisK'
```

Zadanie 2. (1p) Do biblioteki `utf8` dopisz funkcję `normalize(s)` która usuwa z napisu wszystkie znaki nie należące do ASCII.

```
utf8.normalize('Księżyc:\nNów') --> 'Ksiyc:\nNw'
```

Zadanie 3. (1p) Do biblioteki `utf8` dopisz funkcję `sub(s, i [, j])` która działa podobnie jak `string.sub` (możesz pominąć obsługę ujemnych indeksów, ale nie możesz używać `utf8.offset`) dla napisów w UTF-8.

```
utf8.sub('Księżyc:\nNów', 5, 10) --> 'życ:\nN'
```

Zadanie 4. (1p) Do biblioteki `string` dopisz funkcję `strip(s [, w])` która usuwa z końca napisu białe znaki (jeśli wywołana bez argumentów) lub dowolne ze znaków występujące w drugim argumencie. Funkcja może zakładać, że wszystkie znaki mają długość jednego bajtu.

```
string.strip(' test string  ') --> ' test string'
string.strip(' test string', 'tng') --> ' test stri'
```

Zadanie 5. (1p) Do biblioteki `string` dopisz funkcję `split(s [, w])` która rozdziela napis dla zadanego ogranicznika i zwraca powstałą w ten sposób sekwencję. Ogranicznik jest jednoznakowym napisem, domyślnie spacją. Funkcja może zakładać, że wszystkie znaki mają długość jednego bajtu.

```
string.split(' test string  ')
--> {'', 'test', 'string', '', ''}
string.split('test,12,5,,xyz', ',')
--> {'test', '12', '5', '', 'xyz'}
```

Zadanie 6. (2p) Napisz funkcję `lreverser(source, target)`, która wczytuje tekst ze źródła i przepisuje jego linie w odwrotnej kolejności.
Wywołana bez żadnych argumentów powinna czytać ze standardowego wejścia i pisać na standardowe wyjście. Z jednym argumentem (nazwą pliku) powinna wczytywać tekst z tego pliku i wypisywać na standardowe wyjście. Wywołana z dwoma argumentami powinna czytać z pierwszego pliku i pisać do drugiego. Jeśli plik wyjściowy istnieje program powinien poprosić o potwierdzenie jego nadpisania.

Sprawdzenie czy plik istnieje może wyglądać np. tak:

```
local f=io.open(fname,"r")
if f~=nil then
 -- plik istnieje
 io.close(f)
end
```

Zadanie 7. (2p) Przetestuj wydajność kodu Lua przepisującego dane z jednego pliku do drugiego następującymi metodami:

- bajt po bajcie
- linia po linii
- blokami wielkości 8KB
- cały plik na raz

W przypadku ostatniej opcji sprawdź jak duże pliki można tak kopiować.

Do mierzenia czasu skorzystaj z `os.clock`:

```
local x = os.clock()
print ('time till now:', x)
--> time till now:  0.001
local s = 0
for i=1, 100000000 do s = s + i end
print (string.format('elapsed time: %.2f', os.clock() - x))
--> elapsed time: 1.09
```