

I smer, Programske paradigme 2016/2017, kolokvijum, A grupa

Na *Desktop*-u napraviti direktorijum čije je ime u formatu **PPI_kolokvijum_A_ImeIPrezime_BrojIndeksa**. Na primer, **PPI_kolokvijum_A_JovanPetrovic_mi14072**. Sve zadatke sačuvati u ovom direktorijumu. Zadatke imenovati sa **1.py, 2.py, 3.hs, 4.hs**.

NAPOMENA: *eliminacioni test primeri su obeleženi zvezdicom. Potrebno je da se strogo držite formata ispisa koji je naznačen u zadacima.*

1. **[Python]** Datoteke sa ekstenzijom `.json` sadrže podatke o fudbalerima (ime, nacionalnost i broj golova) u sledećem formatu:

```
1 [ { "Ime" : "Alexis Sanchez", "Nacionalnost" : "Cile", "Golovi" : 17 } , ... ]
```

Napisati program koji iz izabrane datoteke izdvaja fudbalere određene nacionalnosti i sortira ih rastuće po broju golova. Željena nacionalnost (npr. 'Engleska') i ime datoteke u kojoj se nalaze podaci o fudbalerima se zadaju kao argumenti komandne linije, a rezultat rada programa se upisuje u datoteku `izabrana_nacionalnost.json` (npr. `Engleska_nacionalnost.json`). U slučaju greške prilikom pokretanja programa, ispisati tekst `Greska` na standardni izlaz.

Primer 1*

```
ARSENAL.JSON
[{"Nacionalnost": "Cile",
 "Ime": "Alexis Sanchez", "Golovi" : 17 },
 {"Nacionalnost": "Francuska",
 "Ime": "Olivier Giroud", "Golovi" : 8 },
 {"Nacionalnost": "Engleska",
 "Ime": "Theo Walcott", "Golovi" : 8},
 {"Nacionalnost": "Engleska",
 "Ime": "Alex Oxlade-Chamberlain",
 "Golovi" : 2},
 {"Nacionalnost": "Francuska",
 "Ime": "Laurent Koscielny", "Golovi" : 2 } ]
POZIV: python 1.py Engleska arsenal.json
ENGLESKA_NACIONALNOST.JSON
[{"Nacionalnost": "Engleska",
 "Ime": "Alex Oxlade-Chamberlain",
 "Golovi" : 2},
 {"Nacionalnost": "Engleska",
 "Ime": "Theo Walcott", "Golovi" : 8}]
```

Primer 2

```
LIVERPOOL.JSON
[{"Nacionalnost": "Senegal",
 "Ime": "Sadio Mane", "Golovi" : 12 },
 {"Nacionalnost": "Engleska",
 "Ime": "Adam Lallana", "Golovi" : 7},
 {"Nacionalnost": "Brazil",
 "Ime": "Roberto Firmino", "Golovi" : 9 },
 {"Nacionalnost": "Brazil",
 "Ime": "Philippe Coutinho", "Golovi" : 6 },
 {"Nacionalnost": "Engleska",
 "Ime": "James Milner", "Golovi" : 5 },
 {"Nacionalnost": "Nemacka",
 "Ime": "Emre Can", "Golovi" : 3 },
 {"Nacionalnost": "Brazil",
 "Ime": "Philippe Coutinho", "Golovi" : 6},
 {"Nacionalnost": "Brazil",
 "Ime": "Roberto Firmino", "Golovi" : 9}]
POZIV: python 1.py Brazil liverpool.json
BRAZIL_NACIONALNOST.JSON
[{"Nacionalnost": "Brazil",
 "Ime": "Philippe Coutinho", "Golovi" : 6},
 {"Nacionalnost": "Brazil",
 "Ime": "Roberto Firmino", "Golovi" : 9}]
```

2. **[Python]** Napisati program koji ređa brojeve u magičan trougao. Magičan trougao je sačinjen od devet brojeva iz intervala [1,9], takvih da svaka stranica sadrži četiri broja koji u zbiru daju 21. Na primer:

```
  3
 1 6
9  5
8 4 2 7
```

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

KOMANDA ISPISA REŠENJA:

```
1 print """
   {0:d}
3  {1:d} {2:d}
   {3:d} {4:d}
5  {5:d} {6:d} {7:d} {8:d}
   """ .format(r['A'], r['B'], r['C'], r['D'], r['E'], r['F'], r['G'], r['H'], r['I'])
```

Okrenite stranu!

3. [Haskell] Napisati funkcije:

(a) `trougao n`

`trougao :: Int -> [(Int,Int,Int)]`

koja kao rezultat vraća listu trojki (x,y,z) takvih da brojevi x , y i z pripadaju intervalu od 1 do n i mogu predstavljati dužine stranica trougla.

UPUTSTVO:

Tri broja mogu predstavljati dužine stranica trougla ukoliko za svaka dva broja važi da im je zbir veći od trećeg broja.

Primer 1*

```
POZIV: trougao 2
IZLAZ:
[(1,1,1), (1,2,2), (2,1,2), (2,2,1), (2,2,2)]
```

Primer 2

```
POZIV: trougao 4
IZLAZ:
[(1,1,1), (1,2,2), (1,3,3), (1,4,4), (2,1,2), (2,2,1), (2,2,2), (2,2,3), (2,3,2),
(2,3,3), (2,3,4), (2,4,3), (2,4,4), (3,1,3), (3,2,2), (3,2,3), (3,2,4), (3,3,1),
(3,3,2), (3,3,3), (3,3,4), (3,4,2), (3,4,3), (3,4,4), (4,1,4), (4,2,3), (4,2,4),
(4,3,2), (4,3,3), (4,3,4), (4,4,1), (4,4,2), (4,4,3), (4,4,4)]
```

(b) `raznostranicni lista`

`raznostranicni :: [(Int,Int,Int)] -> Int`

koja kao argument prihvata listu trojki koje predstavljaju dužine stranica trouglova i vraća broj raznostraničnih trouglova.

UPUTSTVO:

Prilikom poziva funkcije `raznostranicni` koristiti rezultat implementirane funkcije `trougao`.

Primer 1*

```
POZIV: raznostranicni (trougao 2)
IZLAZ:
0
```

Primer 2

```
POZIV: raznostranicni (trougao 4)
IZLAZ:
6
```

4. [Haskell] Grupa hakera je osmislila poseban algoritam za šifrovanje podataka. Podaci predstavljaju listu celih brojeva. Šifrovanje funkcioniše tako što se hakeri dogovore oko tajnog broja k pomoću koga računaju kontrolnu cifru koja se šalje nakon svakih k brojeva. Kontrolna cifra se računa na sledeći način:

- ukoliko je broj k deljiv sa 2, kontrolna cifra se računa kao zbir prethodnih k brojeva
- ukoliko je broj k nije deljiv sa 2, kontrolna cifra se računa kao proizvod prethodnih k brojeva

Napisati funkciju:

`sifruj lista k`

`sifruj :: [Int] -> Int -> [Int]`

koja kao argumente dobija listu brojeva koje hakeri žele da pošalju i broj k , a kao rezultat vraća listu šifrovanih podataka. Podaci se šifruju tako što se nakon svakih k brojeva šalje kontrolna cifra (koja se računa pomoću netransformisanih brojeva), a tih k brojeva se transformiše tako što se pozitivnim brojevima dodaje 1 a negativnim oduzima 1.

Primer 1*

```
POZIV: sifruj [1,-2,3,-4] 2
IZLAZ:
[2,-3,-1,4,-5,-1]
```

Primer 2

```
POZIV: sifruj [-4,14,5,-6,-2,-1,-2,4,11,9,-7,15] 3
IZLAZ:
[-5,15,6,-280,-7,-3,-2,-12,-3,5,12,-88,10,-8,16,-945]
```