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

Na *Desktop*-u napraviti direktorijum čije je ime u formatu **PPI_kolokvijum_B_ImeIPrezime_BrojIndeksa**. Na primer, **PPI_kolokvijum_B_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]** U direktorijumu *utakmice* se nalaze datoteke sa ekstenzijom `.json` koje sadrže podatke o utakmicama (timovi koji se takmiče i vreme početka utakmice) za svaki sport posebno u sledećem formatu:

```
[ { "Timovi": "Liverpool-Arsenal", "Vreme": "18:00"} , ...]
```

   Napisati program koji sa standardnog ulaza učitava podatke o početku i kraju vremenskog intervala u formatu *%H:%M* i iz datoteka učitava podatke o utakmicama za sve sportove, i na standardni izlaz ispisuje listu utakmica čije se vreme početka nalazi u opsegu unešenog vremenskog intervala.

Primer 1*

```
FUDBAL.JSON
  [ {"Timovi": "Liverpool-Arsenal", "Vreme": "18:00"},
   {"Timovi": "Milan-Cheivo", "Vreme": "17:00"},
   {"Timovi": "Eibar-Real Madrid", "Vreme": "18:30"},
   {"Timovi": "Roma-Napoli", "Vreme": "16:15"},
   {"Timovi": "Koln-Bayern", "Vreme": "17:30"} ]
POZIV: python 1.py
ULAZ:
  Unesite početak intervala: 17:00
  Unesite kraj intervala: 18:00
IZLAZ:
   {"Timovi": "Liverpool-Arsenal", "Vreme": "18:00"}
   {"Timovi": "Milan-Cheivo", "Vreme": "17:00"}
   {"Timovi": "Koln-Bayern", "Vreme": "17:30"}
```

Primer 2

```
FUDBAL.JSON
  [ {"Timovi": "Liverpool-Arsenal", "Vreme": "18:00"},
   {"Timovi": "Milan-Cheivo", "Vreme": "17:00"},
   {"Timovi": "Eibar-Real Madrid", "Vreme": "18:30"},
   {"Timovi": "Roma-Napoli", "Vreme": "16:15"} ]
   {"Timovi": "Koln-Bayern", "Vreme": "17:30"} ]
KOSARKA.JSON
  [ {"Timovi": "Orlando-Miami", "Vreme": "18:30"},
   {"Timovi": "Philadelphia-New York", "Vreme": "17:15"},
   {"Timovi": "Atlanta-Cleveland", "Vreme": "19:00"},
   {"Timovi": "Washington-Toronto", "Vreme": "15:30"},
   {"Timovi": "Dallas-Memphis", "Vreme": "16:30"} ]
POZIV: python 1.py
ULAZ:
  Unesite početak intervala: 16:30
  Unesite kraj intervala: 17:30
IZLAZ:
   {"Timovi": "Philadelphia-New York", "Vreme": "17:15"}
   {"Timovi": "Dallas-Memphis", "Vreme": "16:30"}
   {"Timovi": "Milan-Cheivo", "Vreme": "17:00"}
   {"Timovi": "Koln-Bayern", "Vreme": "17:30"}
```

2. **[Python]** Napisati Python program koji dodeljuje različite vrednosti različitim karakterima tako da sledeći izraz bude zadovoljen:

```
    TWO
  * TWO
 + EIGHT
= TWELVE
```

   i sve rezultate ispisuje na standardni izlaz koristeći datu komandu ispisa.

   KOMANDA ISPISA REŠENJA: *(two, eight i twelve su promenljive koje sadrže odgovarajući trocifren, petocifren i šestocifren broj redom, odnosno rešenja date kriptaritmetike)*

```
print "{0:d} * {1:d} + {2:d} = {3:d}".format(two, two, eight, twelve)
```

Okrenite stranu!

3. [**Haskell**] Napisati funkcije:

   (a) `trougaoni n`
   `trougaoni :: Int -> [(Int, Int)]`
   koja kao rezultat vraća listu parova `(i,x)` takvih da broj `i` predstavlja indeks, a broj `x` element na tom indeksu u sekvenci prvih `n` trougaonih brojeva. Trougaoni brojevi se generišu na osnovu sledećeg šablona:

```
Indeks:   1   2     3        4     ...       i
Sekvenca: 1   3     6        10    ...    i(i+1)/2
```

```
          *   *     *         *
             * *   * *       * *
                  * * *     * * *
                           * * * *
```

   Primer 1*

```
Poziv: trougaoni 3
Izlaz:
 [(1,1),(2,3),(3,6)]
```

   Primer 2

```
Poziv: trougaoni 4
Izlaz:
 [(1,1),(2,3),(3,6),(4,10)]
```

   (b) `kvadratni lista`
   `kvadratni :: [(Int,Int)] -> [Int]`
   koja kao argument prihvata listu parova listu parova `(i,x)` takvih da broj `i` predstavlja indeks, a broj `x` element na tom indeksu u listi trougaonih brojeva, i kao rezultat vraca listu kvadratnih brojeva koji se mogu generisati pomoću prosleđene liste trougaonih brojeva. Svaki kvadratni broj se računa kao zbir dva susedna trougaona broja.
   Uputstvo:
   Prilikom poziva funkcije ***`kvadratni`*** *koristiti rezultat implementirane funkcije* ***`trougaoni`***.

   Primer 1*

```
Poziv: kvadratni (trougaoni 3)
Izlaz:
 [4,9]
```

   Primer 2

```
Poziv: kvadratni (trougaoni 4)
Izlaz:
 [4,9,16]
```

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.

   Napisati funkciju:

   `desifruj lista k`
   `desifruj :: [Int] -> Int -> [Int]`

   koja kao argumente dobija šifrovanu listu brojeva koju su hakeri poslali i broj `k`, a kao rezultat vraća dešifrovanu listu brojeva koja ne sadrži kontrolne cifre. Brojevi su šifrovani tako što je svakom pozitivnom broju dodata 1, a svakom negativnom oduzeta 1.

   Primer 1*

```
Poziv: desifruj [2,-3,-1,4,-5,-1] 2
Izlaz:
 [1,-2,3,-4]
```

   Primer 2

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