I smer, Programske paradigme 2019/2020, ispit, JUN1

Na *Desktop*-u se nalazi zip arhiva. Raspakovati je i preimenovati raspakovani direktorijum tako da ime bude u formatu: **ppi_rok_Ime_Prezime_miGGXXX** (npr. **ppi_jun1_Jovan_Petrovic_mi14072**).
Pomenuti direktorijum sadrži IntelliJ IDEA projekat sa već kreiranim izvornim fajlovima za rešenja - ne kreirati nove.
Kompilaciona jedinica koja se ne prevodi se neće pregledati, bez obzira na ozbiljnost greške! Nepoštovanje formata ispisa odnosno imenovanja fajlova i direktorijuma će se kažnjavati sa 10% poena na zadatku odnosno čitavom ispitu. Prolazak eliminacionog test primera, označog zvezdicom, je uslov za pregledanje zadatka ili dela zadatka.

1. **[Python (20p)]** Košarkaškom menadžeru Mišku svakoga dana od raznih klubova stižu zahtevi za novim igračima, pri čemu su dva osnovna kriterijuma za angažovanje igrača njegove godine i pozicija na kojoj igra.

   Informacije o svojim trenutno slobodnim igračima Miško čuva u JSON fajlu, gde za svakog igrača pamti broj dresa (*dres*), ime i prezime (*ime*, *prezime*), visinu (*visina*) i starost (*godine*). Na osnovu visine svakog od igrača Miško određuje poziciju na kojoj on igra, tako što sve igrače koji su visoki barem 2m prodaje kao centre (pozicija 5), igrače koji su niži od 190cm mora prodavati kao plejmejkere (pozicija 1), dok ostale igrače prodaje kao bekove (dakle pozicija 3).

   Napisati Python program koji kao prvi argument komandne linije dobija putanju do JSON datoteke koja sadrži informacije o košarkašima koji su trenutno bez angažmana, dok sa standardnog ulaza dobija informacije *godina* o gornjoj starosti igrača i *pozicija* o željenoj poziciji igrača. Program na standardni izlaz ispisuje visinu najvišeg igrača sa pozicije *pozicija* koji je mlađi od *godina* godina. Ukoliko pozicija nije validna ili ne postoji takav igrač, program na standardni izlaz ispisuje -1.

   Za čuvanje informacija o igračima koristiti klasu *Player* sa poljima (*dres*, *ime*, *prezime*, *visina*, *godine* i *pozcija*). Zadatak urađen bez korišćenja klasa nosi najviše 70% poena.

   Primer JSON fajla (`players.json`):

```
[{"dres":15, "ime":"Markus", "prezime":"Pejdz", "visina":185, "godine":25},
 {"dres":23, "ime":"Vanja", "prezime":"Marinkovic", "visina":197, "godine":24},
 {"dres":45, "ime":"Rade", "prezime":"Zagorac", "visina":206, "godine":26},
 {"dres":11, "ime":"Slavko", "prezime":"Vranjes", "visina":230, "godine":34},
 {"dres":13, "ime":"Petar", "prezime":"Bozic", "visina":189, "godine":36},
 {"dres":1, "ime":"Filip", "prezime":"Covic", "visina":178, "godine":28},
 {"dres":34, "ime":"Darko", "prezime":"Milicic", "visina":212, "godine":33},
 {"dres":36, "ime":"Bogdan", "prezime":"Bogdanovic", "visina":199, "godine":28}]
```

   Primer 1*

```
Poziv: python 1.py players.json
Ulaz:
 34
 5
Izlaz:
 230
```

   Primer 2

```
Poziv: python 1.py players.json
Ulaz:
 24.4
 3
Izlaz:
 197
```

   Primer 3

```
Poziv: python 1.py players.json
Ulaz:
 2
 3
Izlaz:
 -1
```

2. **[Python constraint (15p)]** Čuveni kuvar Gordon Remzi stigao je u Beograd na poslovni sastanak sa ciljem da otvori svoj novi restoran. U sred noći probudio se gladan i rešio je da sam sebi pripremi obrok — proteinsku palačinku. Otišao je do obližnje radnje i shvatio da ima dva problema — radnja nije tako bogata sastojcima, pri tome on u džepu ima samo 10 eura, koje može da zameni po kursu od 117 dinara (iako je zvanični kurs Narodne Banke Srbije 117.52 dinara za euro!).

   Sastojke kupuje u kesicama od po 100 grama, a može da kupi najviše 10 kesica, pri tome hoće da napravi palačinku sa što više proteina, a što manje masti i šećera (količina masti mora biti manja od 500 grama, dok šećera ne sme biti više od 150 grama.

   Koristeći podatke iz naredne tabele izračunati koliko protena može imati najbolja palačinka, pri ograničenjima broja kesica, novca, kao i količine masti i šećera u obroku.

   Na standardni izlaz ispisati **samo** jedan broj koji predstavlja maksimalnu količinu proteina.

Sastojak	Cena (din/kesica)	Broj kesica u radnji	Protein (g/kesica)	Masti (g/kesica)	Šećer (g/kesica)
brašno	30	10	20	30	5
plazma	300	20	15	10	30
jaja	50	7	70	150	2
mleko	170	5	40	32	15
višnja	400	3	23	3	45
nutela	450	9	7	15	68

3. [**Haskell (20p)**] Napisati biblioteku funkcija za rad sa bankarskim računima. Klijentima je potrebno omogućiti otvaranje računa na zahtev, kao i mogućnost uplate na račun.

   Klijentski račun posmatrati kao par `(String, Int)` gde levi element predstavlja broj računa a desni trenutnu količinu novca na računu. Implementirati funkcije:

   - `otvori b br`, čiji je tip: `otvori :: [(String, Int)] -> String -> [(String, Int)]`
     koja otvara račun u banci tako što novi račun sa brojem `br` i inicijalnim iznosom 0 stavlja **na početak** liste računa `b` i vraća potencijalno modifikovanu listu računa (račun se ne otvara ukoliko drugi račun sa istim brojem već postoji u banci)
   - `zatvori b br`, čiji je tip: `zatvori :: [(String, Int)] -> String -> [(String, Int)]`
     koja zatvara račun u banci tako što uklanja račun sa brojem `br` iz liste računa `b` i vraća potencijalno modifikovanu listu računa
   - `uplati b br iznos`, čiji je tip: `uplati :: [(String, Int)] -> String -> Int -> [(String, Int)]`
     koja uplaćuje `iznos` na račun `br` iz liste računa `b` i vraća potencijalno modifikovanu listu računa

   Primer 1*

   ```
   Poziv: otvori [("acc-007", 5), ("acc-005", 6)] "acc-003"
   Izlaz:
     [("acc-003", 0), ("acc-007", 5), ("acc-005", 6)]
   ```

   Primer 2*

   ```
   Poziv: uplati [("acc-007", 5), ("acc-005", 6)] "acc-007" 5
   Izlaz:
     [("acc-007", 10), ("acc-005", 6)]
   ```

   Primer 3*

   ```
   Poziv: zatvori [("acc-007", 5), ("acc-005", 6)] "acc-007"
   Izlaz:
     [("acc-005", 6)]
   ```

4. [**Haskell (bez korišćenja rekurzije) (25p)**] Napraviti biblioteku funkcija za rad sa bankarskim transakcijama. Svaka transakcija se sastoji od identifikatora, iznosa i brojeva računa pošiljaoca i primaoca. U okviru biblioteke je neophodno isporučiti tip podataka koji predstavlja transakciju, kolekciju izvršenih transakcija, kao i funkcije za listanje, izvršavanje i poništavanje transakcija.

   Implementirati (bez rekurzije, koristeći funkcije višeg reda):

   - prikazivi tip podataka `Transakcija` (podrazumevano instancira `Show` i `Eq`) sa konstruktorom `MkTransakcija` i poljima: `ident` (identifikator, ceo broj, jedinstven za svaku transakciju), `iznos` (ceo broj), `posiljalac` (niska koja predstavlja broj računa pošiljaoca) i `primalac` (niska koja predstavlja broj računa primaoca)
   - tip podataka `AktivneTransakcije` koji predstavlja alias za kolekciju transakcija
   - funkciju `izlistaj ts br`, čiji je tip: `izlistaj :: AktivneTransakcije -> String -> [Transakcija]`
     koja vraća sve transakcije iz aktivnih transakcija (`ts`) u kojima figuriše dati broj računa (`br`), bilo kao pošiljalac ili primalac
   - funkciju `dodaj ts t`, čiji je tip: `dodaj :: AktivneTransakcije -> Transakcija -> AktivneTransakcije`
     koja dodaje transakciju (`t`) **na početak** spiska aktivnih transakcija (`ts`) i vraća modifikovani spisak transakcija
   - funkciju `ukloni ts ident`, čiji je tip: `ukloni :: AktivneTransakcije -> Int -> AktivneTransakcije`
     koja uklanja transakciju sa datim identifikatorom (`ident`) iz spiska aktivnih transakcija (`ts`) i vraća modifikovani spisak transakcija
   - funkciju `ukupno ts`, čiji je tip: `ukupno :: AktivneTransakcije -> Int`
     koja računa ukupni iznos svih transakcija

5. [**Scala Spark (20p)**] Napisati program za parsiranje polisa osiguranja. Potrebno je pronaći centroid koordinata svih rezidencijalnih polisa (tipa `Residential`) gde je konstrukcija tipa `Wood` u okrugu `PALM BEACH COUNTY`.

   U datoteci `insurance.csv` se nalaze podaci u *CSV* formatu:

   ```
   ID,kod_drzave,okrug,geo_sirina,geo_duzina,tip_polise,konstrukcija
   ```

   Primer linija u datoteci:

   ```
   ID,kod_drzave,okrug,geo_sirina,geo_duzina,tip_polise,konstrukcija
   119736,FL,CLAY COUNTY,30.102261,-81.711777,Residential,Masonry
   524433,FL,SUWANNEE COUNTY,29.962601,-82.926155,Residential,Wood
   206893,FL,CLAY COUNTY,30.089579,-81.700455,Residential,Wood
   985870,FL,PALM BEACH COUNTY,26.694948,-80.070671,Commercial,Reinforced Concrete
   ```

   Primer izlaza:

   ```
   (25.123456, -85.123456)
   ```

   Voditi računa o skupoći upita. Nije dozvoljeno korišćenje `foreach` funkcije.

   Uputstvo: *Centroid je tačka u 2D ravni takva da je njena* x *(y) koordinata prosek* x *(y) koordinata svih tačaka.*