Domaći zadatak 4

1. Data je lista stringova ["flower","flow","flight"]. Napisati funkciju koja koristi reduce da nađe najduži string među datim stringovima.

2. Date su dvije liste, jedna sa imenima studenata, a druga sa njihovim prosječnim ocjenama. Napisati funkciju koja pronalazi studente sa prosječnom ocjenom iznad 8.5 i vraća listu torki (ime, ocjena) za te studente.

3. Data je lista rječnika oblika [{ 'ime': 'Ana', 'godine': 22, 'prosek': 9.1 }, ...]. Napisati funkciju koja filtrira studente starije od 21 godine i sortira ih po prosjeku, koristeći filter, sort i lambda funkcije.

4. Data je lista ["Hello, World!", "Python is cool", "Functional programming rocks"]. Napisati funkciju koja broji ukupan broj riječi u svim stringovima koristeći map i reduce.

5. Data je lista torki oblika [(ime, ocjena, predmet), ...]. Napisati funkciju koja koristi filter, map, i reduce da izračuna prosječnu ocjenu po predmetu.

6. Dat je niz vrijednosti koji predstavlja vremensku seriju. Napisati funkciju koja koristi map da izračuna promjenu (diferenciju) između svakog uzastopnog para vrednosti.

7. Data je lista ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']. Napisati funkciju koja koristi map i reduce da izračuna frekvenciju svakog elementa.

8. U fajlu link nalaze se dimenzije pravougaonika (svaki red u fajlu je kombinacija stranica pravougaonika a i b, vrijednosti su razdvojene zarezom). Pretpostaviti da se uvijek radi o prirodnim brojevima. Vaš zadatak je da iz fajla izdvojite sve pravougaonike koji su kvadrati (a = b), i da štampate površinu najvećeg kvadrata. Za ovaj primjer output bi trebao da bude 16 (kvadrat sa najdužim stranicama je poslednji, čija je dužina stranica 4).

9. Napisati program koji za unijeti naziv grada iz fajla cities.txt štampa naziv naselja sa najvećim brojem stanovnika. Fajl u svakom redu sadrži string oblika grad, naziv_naselja, broj_stanovnika
   Input: Podgorica **Output**: ‘Zabjelo’
   Fajl:
   Podgorica,Zabjelo,30000
   Niksic,Ozrinici,2500
   Podgorica,Donja Gorica,20000

10. Napisati program koji za unijeti naziv države iz fajla population.txt štampa grad sa najmanjim brojem stanovnika. Fajl u svakom redu sadrži string oblika: Drzava,Grad,broj_stanovnika

    Input: Poljska **Output**: 'Varsava'

    Fajl:

    Poljska,Krakov,1725000
    Njemacka,Berlin,3769000
    Poljska,Varsava,1708000

11. Napisati program koji za unijeti naziv države i zadati fajl population.txt štampa ukupan broj stanovnika za zadatu državu. Fajl u svakom redu sadrži string oblika: Drzava,Grad,broj_stanovnika

    Input: Poljska **Output**: 3433000

    Fajl:

    Poljska,Krakov,1725000
    Njemacka,Berlin,3769000

12. Napisati program koji iz fajla (kreirajte txt fajl) izvlaci sve heksadecimalne brojeve (pocinju sa 0x), konvertuje ih ih brojeve sa osnovom 10(int) i ako se nakon konverzije u int zavrsavaju sa cifrom 3, dodati ih u sumu.

    Ulaz (fajla sa hexa zapisima):

    0xA
    22
    ABC
    0x3
    0xC
    121
    0xD
    0x17

    Izlaz: 39 (jer se 0x3, 0xD, i 0x17 u heksadecimalnom zapisu zavrsavaju sa brojem 3, pa je 3 + 13 + 23 = 39)

13. Sledeće zadatke potrebno je implementirati kao posebne funkcije:
    - Kreirati funkciju append_to_file koja ima jedan parametar list_of_products (lista proizvoda gdje je svaki proizvod dictionary oblika {“naziv”: ime_proizvoda, “opis”: opis_proizvoda, “godina”: godina_proizvodnje, “kolicina”: broj_dostupnih_proizvoda, “cijena”: cijena_proizvoda}), a dodaje proizvode u fajl products.csv (pretpostaviti da fajl već postoji i da je prazan) i to u formatu: Naziv, Opis, Godina, Kolicina, Cijena. Pretpostaviti da su unosi ispravni (ne treba raditi validaciju)

      Primjer:

      Input:

      [{“naziv”: ”Televizor”, “opis”: ”LG televizor 43inc”, “godina”: 2019, “kolicina”: 10, “cijena”: 300}, {“naziv”: ”Televizor”, “opis”:” Samsung televizor 39inc”, “godina”: 2017, “kolicina”: 5, “cijena”: 250},]

      Output: (products.txt):

      Naziv, Opis, Godina, Kolicina, Cijena

      Televizor, LG televizor 43inc, 2019, 10, 300

      Televizor, Samsung televizor 39inc, 2017, 5, 250
    - Kreirati funkciju get_products_older_than koja ima jedan parametar i to godinu koja predstavlja najstarije proizvode koje želite da izdvojite iz fajla.

      Input: 2018 (izdvoji sve proizvode iz fajla products.csv čija je vrijednost ključa “godina” veća ili jednaka godini 2018)

      Output (ako posmatrate fajl products.csv iz primjera pod a): Televizor, LG televizor 43inc, 2019, 10, 300
    - Kreirati funkciju max_possible_revenue koja računa koliki je maksimalni prihod ako se proda svaki proizvod (posmatrati fajl products.csv)

      Input: Svi proizvodi učitani kao lista dictionary-a iz fajla products.csv

      Output: 4250 (10 * 300 + 5 * 250)

14. Sledeće zadatke potrebno je implementirati kao posebne funkcije:
    - Kreirati funkciju append_to_file koja ima jedan parametar list_of_students (lista studenata gdje je svaki student dictionary oblika {“ime”: ime_studenta, “prezime”: prezime_studenta, “godina”: godina_studija, “prosjek”: prosjek_studenta}), a dodaje studente u fajl students.txt (pretpostaviti da fajl već postoji i da je prazan) i to u formatu: Ime,Prezime,godina,prosjek. Napomena: Pretpostaviti da su unosi ispravni (ne treba raditi validaciju), da je godina studija između 1 i 8, a grade od 6 do 10 (mogu da budu i decimalni brojevi)

      Primjer:

      Input:

      [{“ime”: ”Marko”, “prezime”:”Markovic”, “godina”: 2, “prosjek”: 8.6 }, {“ime”: ”Boris”, “prezime”:”Boricic”, “godina”: 3, “prosjek”: 7.9 }, {“ime”: ”Novak”, “prezime”: ”Novovic”, “godina”: 3, “prosjek”: 6.9 }]

      Output: (students.txt):

      Marko,Markovic,2,8.6

      Boris,Boricic,3,7.9

      Novak,Novovic,3,6.9
    - Kreirati funkciju get_students_with_greater_grade koja ima dva parametra year i grade, gdje je year cio broj između 1 i 8 i predstavlja godinu studija, a grade A (9.5 - 10), B (8.5 - 9.5 ne uključujući), C (7.5 - 8.5 ne uključujući), D (6.5 - 7.5 ne uključujući) ili E (6 - 6.5 ne uključujući). Funkcija treba da izdvoji one studente čija je ocjena veća ili jednaka u odnosu na onu koju ste unijeli kao argument. Studente treba vratiti kao niz dictionary elemenata (kao što su unošeni)

      Input: 3, C (izdvoji sve studente sa treće godine čiji je prosjek >= 7.5)

      Output (ako posmatrate fajl student.txt iz primjera pod a): [{“ime”:”Boris”, “prezime”:”Boricic”, “godina”: 3, “prosjek”: 7.9 }]
    - Sve funkcije je potrebno testirati sa bar dvije različite kombinacije ulaznih argumenata funkcija

15. Vaš zadatak je da napišete program za validaciju broja kreditne kartice. Broj cifara broja kartice je 16 (treba odraditi validaciju da unos samo sadrži cifre i da je dužina stringa tačno 16). Algoritam je sledeći:
    - Potrebno je duplirati svaku drugu cifru i sačuvati vrijednost (počevši sa desna u lijevo)
    - Ako nakon dupliranja dobijete broj veći od 9, potrebno je sumirati sve cifre broja (npr. ako duplirate 7, dobićete 14, ali taj broj treba transformisati u 1 + 4, tj. 5)
    - Nakon toga potrebno je odraditi sabiranje svih cifara broja, a onda dobijeni broj podijeli sa 10.
    - Ukoliko ne dobijete ostatak, kreditna kartica je validna

      Primjer (samo dio cifara prikazan):

      12345 ⇒ [1, 2*, 3, 4*, 5] ⇒ [1, 4, 3, 8, 5]

      1234 ⇒ [1*, 2, 3*, 4] ⇒ [2, 2, 6, 4] (ako vas ovo zbunjuje možete da okrenete broj, pa da kvadrirate svaki drugu cifru)

      891 ⇒ [8, 9*, 1] ⇒ [8, 18, 1] ⇒ [8, 1+8, 1] ⇒ [8, 9, 1]

      b) Iskoristiti funkciju koju ste kreirali za validaciju kreditnih kartica koju ste implementirali u prvom dijelu zadatka. Potrebno je da sada prođete kroz sve brojeve kartica koji se nalaze u fajlu. Kreirati novi fajl koji će da sadrži dodatnu kolonu koja označava da li su brojevi kartica validni ili ne. (**Valid** ili **Invalid**)

16. (Dodatni) Kreirati fajl igrice.txt u kome se svaki igra, pojedinačno, čuva u jednom redu, tj. ako imate unos od 5 igara, fajl treba da sadrži 5 linija. Fajl treba da sadži bar 10 igara. Igre prvo unosite ručno (ne pomoću koda).

    Svaka igra je opisana:

    nazivom, ocjenom, godinom izlaska, izdavačem i žanrovima

    Atributi igre odvojeni su sa ; a za njih važi sledeće:

    - **naziv** predstavlja string dužine od 2 do 50 karaktera
    - **ocjene** su zaokruženi float brojevi (od 1 do 10) na dvije decimale
    - **godina** je prirodan broj veći 1950, a manji od tekuće godine
    - **izdavač** string od 2 do 40 karaktera (nije obavezno)
    - **žanrovi** niz stringova odvojenih sa space, igra može da ima maksimalno 3 žanra

    Primjer kako jedna igra treba da bude sačuvana:

    GTA 5;9.5;2012;Rockstar;Action Crime

Potrebno je prvo provjeriti da li su unijete igre ispravno unesene (npr. ako je naziv filma duži od 50 ili kraći od dva karaktera to je neispravan unos), tj. filtrirati ih.
Napomena: Žanrovi su unaprijed definisati i čuvaju se u fiksnoj listi. Lista treba da sadži bar 5 žanrova.

Nakon filtriranja u terminalu prikazati igre, a osim toga omogućiti korisniku, tj. pitati korisnika da li želi da unese nove igre ili ne. Ne dozvoliti neispravan unos da ne bi moralo da se radi novo filtriranje fajla. Prikazati odgovarajuću grešku u slučaju pogrešnog unosa i tražiti od korisnika da ponovo unese igre ili da prekine unos. Nove igre se upisuju u fajl obavezno u ispravnom formatu sa validnim vrijednostima atributa.
Nakon filtriranja i eventualnog unosa novih igara iz novog fajla potrebno je da učitate sve igre (u tom fajlu igre bi trebalo da su pravilno unesene). Svaku igru potrebno pročitati iz fajla i sačuvati je kao dictionary u listi igara, tj. Kreirati listu igara gdje je svaka igra dictionary oblika:
{naziv:string,ocjena:float_broj, godina:int_broj, izdavac:string zanrovi:lista_stringova}

Omogućiti korisniku da radi filtriranje po bilo kom atributu igre i to na sledeći način:

- Po nazivu (igre koje počinju sa zadatim termom)
- Po ocjeni (igre čija je ocjena veća od zadate)
- Po godini (može da odabere da li želi da traži filmove prije ili nakon unijete godine)
- Po izdavaču (igre koje počinju sa zadatim termom za izdavača)
- Po žanru (moguće je pretraživanje po jednom, dva ili tri žanra)

Korisniku pri unosu napomenuti da su ocjene od 1 do 10, a osim toga treba napomenuti i koje žanrove može da odabere. Ako korisnik unese podatke pogrešno, treba da mu se prikaže greška i da mu se napomene da opet unosi novi input. Nakon tog filtriranja treba prikazati rezultate/igre u terminalu.

17. Napisati **dekorator** koji koristi funkcionalni pristup za mjerenje vremena izvršavanja funkcije i primenite ga na nekoliko različitih funkcija.