Zadaci za vježbanje

1. Narcissistic Number je broj čija suma cifara (tog broja) stepenova sa njegovim brojem cifara daje isti taj broj.

   Primjer 1: 153 (3 cifre)
   1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153
   Primjer 2: 1634 (4 cifre):
   1^4 + 6^4 + 3^4 + 4^4 = 1 + 1296 + 81 + 256 = 1634
   Vaš program treba da vrati **true** ili **false** u zavisnosti od toga da li je broj Narcissitic ili nije. Input je uvijek validan broj.

2. Napisati program koji provjerava da li se zadati broj nalazi u zadatom segmentu.
   Primjer: *ran_inclusive*(3, 10, 5) vraća **true** jer je 3 <= 5 <= 10,
   ran_inclusive (-10, 13, -25) vraća **false** jer je -25 manji od -10, a samim tim i od 13, pa nije iz zadatog segmenta

3. Napisati program koji za unijeti URL (string), izvlači (parsira) samo domain name i vraća ga kao string. Pretpostaviti da korisnik unosi ispravan URL.

   Primjeri:
   get_domain("http://github.com/carbonfive/raygun"), izlaz "github.com"
   get_domain("https://google.com"), izlaz "google.com"
   get_domain("http://github.com/carbonfive/raygun"), izlaz "github.com"
   get_domain("http://www.zombie-bites.com"), izlaz "zombie-bites.com"

4. Dječakov put od škole do kuće je dug. Da bi mu bilo interesantnije, odlučio je da sabira sve brojeve kuća (na svakoj kući piše adresa, tj. broj) pored kojih prođe dok ide do kuće. Nažalost, nemaju sve kuće brojeve na njima, a osim toga dječak redovno mijenja ulice, tako da se brojevi ne pojavljuju u nekom definisanom redosledu. U jednom momentu tokom šetnje, dječak naiđe na kuću na kojoj piše 0, što ga je iznenadilo toliko da je zaboravio (prestao) da sabira brojeve nakon što je naišao na ovu kuću. Za zadati niz kuća (svaka identifikovana sa brojem) odrediti zbir koji je dječak dobio.
   Primjer:
   Za **input** = [5, 1, 2, 3, 0, 1, 5, 0, 2], **output** treba da bude 11 (5 + 1 + 2 + 3 = 11)

5. Klijenti postavljaju zahtjeve brokeru za kupovinu/prodaju akcija. Zahtjevi mogu da budu jednostavni ili višestruki (više jednostavnih). Zahtjev ima sledeći format:

   Quote /space/ Quantity /space/ Price /space/ Status

   gdje **Quote** predstavlja naziv akcije, sadrži non-whitespace karaktere,
   Quantity je prirodan broj koji predstavlja broj akcija koje se prodaju/kupuju, **Price** je float koji predstavlja cijenu pojedine akcije (sa decimalnom tačkom "."), **Status** je B (buy) ili S (sell) koji predstavlja da li se akcije prodaju ili kupuju.

   Primjer 1 (simple):
   "GOOG 300 542.0 B"

   Višestruki zahtjevi se sastoje od više simple zahtjeva koji su spojeni zarezom

   Primjer 2 (multiple-višestruki):
   "ZNG 1300 2.66 B,NY 50 56.32 B,OWW 1000 11.623 B,OGG 20 580.1 B"

   Da olakšate brokeru posao vaš zadatak je da mu vratite string "Buy: b Sell: s" gdje su b i s formata 'double' zaokruženog na 2 decimalse, b predstavlja ukupnu cijenu kupljenih akcija, a s ukupnu cijenu prodatih akcija.

   Output za primjer 2:
   "Buy: 29499.00 Sell: 0"

6. Vaš program treba da nađe najdužu sekvencu izastopnih nula za unijetu listu. Takodje, treba da vrati pocetnu i krajnju poziciju te podliste u listi

   Primjer:
   Niz [1, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0] ima tri sekvence uzastopnih nula sa dužinama 3, 1 i 4.

   Vraća niz [4, 7, 10] gdje je 4 duzina podniza, 7 startna pozicija (uključujući), 10 krajnja pozicija (ukljucujući)

7. Napisati funkciju koja za zadati string i slovo vraća sve riječi koje se završavaju sa zadatim slovom, indekse zadatog slova, kao i broj riječi koje se završavaju sa zadatim slovom u rečenici.

   Primjer: *get_words_ends_with_letter* (“Print only the words that end with the chosen letter in those sentences. Example can contains one or more sentences.”, “s”)

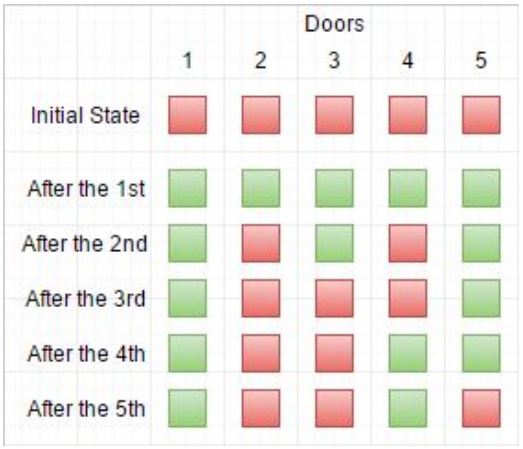
   vraća niz objekata sledećeg oblika:
   [{ { word: “words”, position: 19 }, { word : “sentences”, position : 70 }, num_of_words: 2 }, { { word: “contains”, position: 92}, { word: “sentences”, position: 114 }, num_of_words: 2}]

   Objašnjenje: objekti unutar liste predstavljaju informacije o svakoj rečenici pojedinačno. Objekti u rečenici opisuju: key **word** je riječ koja se završava sa zadatim slovom (u primjeru je to slovo s), a key **position** predstavlja indeks slova u unešenom stringu. Key **num_of_words** (*u obije rečenice je to 2*) predstavlja broj riječi koje se u rečenici završavaju sa odabranim slovom.

8. Napisati funkciju koja vraća broj cifara u stringu i kreira od njih integer. **Primjer**: **get_digits**(“Hi Mr. Rober53. How are you today? Today is 08.10.2019”), vraća 5308102019 i to kao integer. **Pomoć**: da provjerite da li je karakter slovo koristiti isalpha metod.

9. Napisati funkciju koja vraća broj malih i broj velikih slova za zadati string.
   Primjer: **upper_lower** (“Hi Mr. Rober. How are you today?”), vraća torku (19, 4), 19 - broj malih slova, 4 - broj velikih slova. Koristeći dobijeni torku izračunati ukupan broj malih i velikih slova. **Pomoć**: da provjerite da li je karakter slovo koristiti isalpha metod.

10. Svakog jutra sva vrata škole su zatvorena. Škola je prilično velika, ima N vrata. Učenici su počeli da dolaze. Teško je za povjerovati, ali svi oni žele da uče. Škola ima N učenika, a oni dolaze jedno po jedno. Kada dijete prođe kroz vrata, ono izmijeni status za vrata (Open-> Closed, Closed-> Opened).
Svaki učenik ima svoj broj, i svaki i-ti mijenja status i-tim vratima. **Na primjer**: kada prvi učenik dođe u školu, on mijenja status svim prvim vratima (otvara ih sve). Drugi mijenja status za svaka druga vrata (druga, četvrta, šesta, itd.). Treći mijenja status za svaka treća vrata (treća, šesta, itd.). Konačno, zadnji učenik (n-ti), mijenja status za svaka n-ta vrata (samo su jedna takva, zadnja).
Vaš zadatak je da izračunate koliko vrata će ostati otvoreno nakon što dođu svi učenici.
Primjer:



Crveni kvadrati – zatvorena vrata, zeleni – otvorena vrata.
Input: n – broj vrata i učenika, n ∈ N, n ∈ [1, 100000]
Output: o – broj otvorenih vrata, o ∈ N
doors(5) treba da vrati 2

11. Vaš zadatak je da napravite **password validator**. Ovaj validator treba da funkcioniše za razne slučajeve i to na osnovu toga kako definišete određene parametre funkcije. Parametri koji mogu da budu True ili False su:
    a. **flagUpper** kojim provjeravate da li string ima ili ne bar jedno veliko slovo
    b. **flagLower** kojim provjeravate da li string ima ili ne bar jedno malo slovo
    c. **flagDigit** kojim provjeravate da li string sadrži bar jednu cifru

    Osim ova tri parametra (koji su postavljeni na False ako se ne definišu), treba proslijediti i minimalnu dužinu stringa koja mora biti zadovoljena, kao i sami string koji validirate. Funkciju treba da izgleda **check_password**(input_string, min_string_len, flagUpper, flagLower, flagDigit)

    Primjer:
    Input input_string = “Passw123”, **output** check_password(input_string, 10, True, True, False) - > False jer smo stavili da je minimalna dužina stringa 10, a u našem primjeru je 8, dok check_password(input_string, 8, True, True, False) - > true što znači da su svi uslovi validacije ispunjeni.

12. Vaš zadatak je da napišete program za validaciju broja kreditne kartice. Broj cifara broja kartice je 16 (treba odraditi validaciju da unos samo sadži cifre i da je dužina stringa tačno 16). Algoritam je sledeći:
    a. Potrebno je duplirati svaku drugu cifru i sačuvati vrijednost (počevši sa desna u lijevo)
    b. 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)
    c. Nakon toga potrebno je odraditi sabiranje svih cifara broja, a onda dobijeni broj podijeli sa 10.
    d. 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]

13. Napisati funkciju koja ima dva parametra

    Parameter 1: HTML kod (string) koji se nalazi između (“ “), npr. :

```
htmlString1 = `
<article id="animals">

  <h1 class="main-heading">Nature's Wonders</h1>
  <p>In this article we discuss animals.</p>

  <section id="birds">
    <h2 class="favourite">Birds</h2>
    <p>
      Forest is a wonderful place to see birds.
    </p>
  </section>

  <section id="butterflies">
    <h2>Butterflies</h2>
    <p>
      Butterflies possess some of the most striking colour displays found in nature.
    </p>
  </section>

</article>`
```

    Parameter 2: String koji predstavlja ime HTML taga, na primjer: 'h2'

    Output: Niz stringova koji predstavljaju sadržaj između otvorenog i zatvorenog taga koji je definisan kao drugi parameter funkcije.

    Primjer 1: *getTagContent*(htmlString1, 'h1')
    Output: [*"Nature's Wonders"*]
    Vodite računa da HTML tag može da sadrži atribute

    Primjer 2: *getTagContent*(htmlString1, 'h2')
    Output: [*"Birds","Butterflies"*]

    Primjer 3: *getTagContent*(htmlString1, 'p')
    Output: [*"In this article we discuss animals.", "Forest is a wonderful place to See birds.", "Butterflies possess some of the most striking color displays found in nature."*]

14. Potrebno je kreirati program za igru Hangman. Koristi se engleski alfabet i engleske riječi. Korisnik unosi slova. Lista riječi koje se mogu uzeti u razmatranje se nalaze u listi koju vi definišete. Prije početka igre, odabrati random riječ iz te liste. Svaki put korisniku treba prikazati koja slova i na kojim pozicijama je pogodio, kao i koliko mu je preostalo poteza prije Game Over.

15. Izračunati za koji dan se desio najveći skok, a za koji dan najveći pad akcija, za 2014. godinu. Porediti akcije na otvaranju i zatvaranju berze za svaki dan i naći min i max. Posmatra se procentualna razlika ovih vrijednosti, a ne brojčana razlika između vrijedosti. CSV fajl koji obrađujete preuzeti sa sledećeg linka. Rezultate prikazati kao dictionary/objekat sledećeg oblika:
{ “**biggest_perc_jump”**:
{“date”: <datetime>, “open”: <float>, “close”: <float>, “difference”: <float>}
“**biggest_perc_drop”**:
{“date”: <datetime>, “open”: <float>, “close”: <float>, “difference”: <float>}
}

16. Za ovaj zadatak potrebno da ja koristite fajl koji se nalazi na sledećem linku. Potrebno je obaviti sledeće operacije nad fajlom:
    a. Izdvojiti sve kategorije iz fajla (izdvojite kolonu “Category” i iz nje samo izdvojite unique elemente, koji tip podatka smo koristili za to)
    b. Nakon toga za svaku kategoriju kreirati poseban fajl, a naziv fajla treba da bude sledeceg formata: **gpstore_categoryname** (gdje je categoryname varijabilno u odnosu na naziv kategorije). Sve fajlove cuvati u odvojenom folderu, categories
    c. Sledeće što treba da uradite je da iz glavnog CSV fajla izdvojite sve aplikacije za koje važi da je broj instalacije veći od 100000 (moraćete da ispravno konvertujete stringove iz kolone Installs u cijele brojeve), a da se zadnji update desio 2018. godine (kolona Last Updated). Rezultate upisati u novom CSV fajlu **filtered_by_noi_and_lu**
    d. Sada trebate da izdvojite sve **Paid** igre čija je cijena ispod 10$, a broj Reviews veći od 10000. U novom fajlu čiji naziv treba da bude **paid_price_less_then_10** sačuvati sve kolone kao iz originalnog CSV fajlva, osim kolone Type gdje se nalaze informacije da li je aplikacija free ili paid

17. Potrebno je kreirati klasu **Article** koja je opisana sledećim
    a. Atributima (svi su private):
        i. **title** (samo alfabetski karakteri, **unique - ne u setteru nego pri kreiranju novog Article**, do 50)
        ii. **author** (samo alfabetski karakteri, ime i prezime autora, ne duže od 100 karaktera ukupno)
        iii. **description** (opis, do 300 karaktera)
        iv. **category** (kategorija, ne više od 20 karaktera)
        v. **views** (broj pregleda, default = 0)
        vi. **comments** (default prazna lista, jedan ili više komentara koji se čuvaju kao niz/lista torki, gdje je prvi element **title** (dužina stringa ne veća od 50 karaktera), drugi element **author** (dužina stringa ne veća od 50 karakera) i **description** (sami sadržaj komentara, broj karaktera string ne veći od 120)
    b. Metodama:
        i. Konstrukor koji kreira objekat Article
        ii. Geteri i seteri za **title**, **author**, **description**, **category views**
            1. Pri kreiranju settera odraditi validaciju koja je gore navedena. Ako validacija ne prođe uspješno, korisniku štampati adekvatnu grešku. Obratiti posebno pažnju na title, jer ne mogu da postoje dva članka (news) sa istim **title**
        iii. **insert_new_comment** (*title, author, description*)
            1. Omogućava dodavanje komentara. Prije dodavanja komentara odraditi adekvatnu validaciju (title unique).
            2. Ako je unos ispravan u listu se dodaje nova torka oblika (title, author, description). Napomena: author nije obavezan, tako da ako ga korisnik ne unese, default vrijednost je “anonim”
        iv. **delete_comment_by_title** (*title*)
            1. Briše komentar u zavisnoti od vrijednosti title parametra
        v. **delete_comments_by_author** (*author*)
            1. Briše sve komentare u zavisnoti od vrijednosti author parametra
        vi. **get_comment_by_title** (*title*)
            1. Vraća komentar u zavisnosti od title u obliku { title: [author, description] }
        vii. **get_comments_by_author** (author)
            1. Vraća sve komentare kao listu torki u zavisnosti od author (torka izgleda kao pri kreiranju)\

viii. **inc_views** (num)
    1. Povećava broj pregleda za zadati broj, ako se ne proslijedi num, broj pregleda se uvećava za jedan (default)
ix. **print_article**
    1. Štampa Article u formatu:

title: string, author: string, description: string, category: string, views: int, number_of_comments: int

18. Kreirati klase **TechArticle** koja proširuje osnovnu klasu **Article**
    a. Dodatni atributi (private) u odnosu na osnovu klasu (koristiti **super**)
        i. **creation_date** (string oblika d/m/y, pretpostaviti da je unos ispravan, tačno 10 karaktera, ako je **d** ili **m** manje od 10 dodaje se 0 kao prefiks, npr. 07/06/2019)
        ii. **lang** (string od tačno dva karaktera, dozvoljene vrijednosti en i rs)
        iii. **Napomena**: category ima default vrijednost “tech”, što znači da pri kreiranju TechArticle korisnik ne unosi kategoriju
    b. Dodatni metodi
        i. Kreirati dodatne getere i setere (za creation date i lang)
            1. Voditi računa o validaciji
        ii. **get_comments_by_term** (term)
            1. Vraća sve komentare čija vrijednost title počinje sa term i to kao listu torki (torka izgleda kao pri kreiranju)
        iii. **print_article**
            1. Štampa Article u formatu:

title: string, author: string, description: string, category: string, views: int, number_of_comments: int, creation_date: string, language: string

19. Za ovaj zadatak potrebno je implementirati dodatne operacije za jednostruko olanačne liste (iskoristi kod sa interneta za neophodne operacije za implementaciju navedenih operacija)
    a. Potrebno je implementirati metod **get_middle_node**(self) koji vraća element sa srednje pozicije u listi

       Primjer:
       Input: 1 -> 2 -> 3 -> 4 -> 5 -> None, **output**: 3
       Input: 1 -> 2 -> 3 -> 4 -> None, **output**: 2 (ako imate paran broj elemenata srednji element je mid - 1 ili mid + 1, vi uzmite da je to mid - 1, koristiti // za dobijanje srednjeg elementa)

    b. Potrebno je implementirati metod **__eq__**(self, other) koji provjerava da li su dvije liste iste. Dvije liste su iste ako sadrže iste vrijednosti na odgovarajućim pozicijama

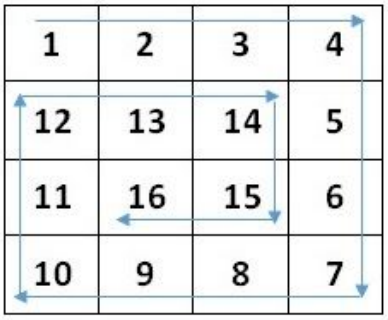
    c. Potrebno je implementirati metod **remove_duplicates**(self) koji uklanja duple elemente iz liste, a čuva prvo pojavljivanje tog elementa

       Primjer:
       Input: 1 -> 2 -> 2 -> 3 -> 1 -> 10, **output**: 1 -> 2 -> 3 -> 10

20. Potrebno je napraviti metod **exchange**(self) koji mijenja prvi i poslednji element kružne liste
    Input: 1 -> 2 -> 3 -> 4 -> 5 -> 1 (1 je head element)
    Output: 5 -> 2 -> 3 -> 4 -> 1 -> 5 (5 treba da postane head element)

21. Potrebno je implementirate sledeće metode za **BinaryTree** (proširiti klasu BinaryTree koja ima osnovne operacije dodavanje/brisanje/stampa cvorova stabla, sve mozete naci na internetu):
    a. Nalazi i štampati najmanji element binarnog stabla
    b. Računa i štampa proizvod dva najmanja čvora u binarnom stablu (za dva čvora se smatra da su najmanja, ako su im vrijednosti najmanje)
    c. Računa i štampa sumu svih elemenata binarnog stabla čija je vrijednost čvora veća od 10
    d. Računa i štampa visinu binarnog stabla

22. Napisati program koji:
    a. Računa proizvod neparnih brojeva sa dijagonale kvadratne matrice
    b. Kreira transponovanu matricu od zadate kvadratne matrice
    c. Formira kružnu kvadratnu matricu za zadatu dimeniziju n, tj. matricu sledećeg oblika



    d. Sabira dvije zadate matrice i vraća novu matricu

23. Naravno da možete da nađete još puno zanimljivih zadataka na:
    a. freeCodeCamp (JS Algorithms)
    b. codeWars
    c. HackerRank
    d. Coding games