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”)

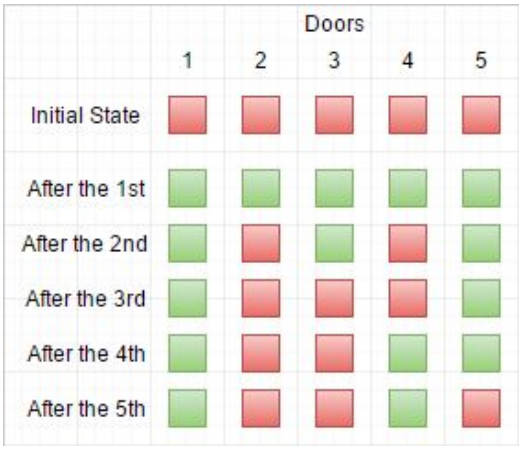
   vraća niz objekata sledećeg oblika:
   [{ { word: “words”, position: 19 }, { word : “sentences”, position : 70 }, num_of_words: 2 }, { { word: “contains”, position: 92}, { word: “sentences”, position: 114 }, num_of_words: 2}]

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