XT – Assesment-1

Duration *: 1 hour*

Ex-1: Non-unique Elements

You are given a non-empty list of integers (X).

For this task, you should return a list consisting of only the non-unique elements in this list.

To do so you will need to remove all unique elements (elements which are contained in a given list only once).

When solving this task, do not change the order of the list.

Example: [1, 2, 3, 1, 3] 1 and 3 non-unique elements and result will be [1, 3, 1, 3].

Input: A list of integers.

Output: An iterable of integers.

Example:

nonUniqueElements([1, 2, 3, 1, 3]) == [1, 3, 1, 3]

nonUniqueElements([1, 2, 3, 4, 5]) == []

nonUniqueElements([5, 5, 5, 5, 5]) == [5, 5, 5, 5, 5]

nonUniqueElements([10, 9, 10, 10, 9, 8]) == [10, 9, 10, 10, 9]

Ex-2: The Most Wanted Letter

You are given a text, which contains different english letters and punctuation symbols.

You should find the most frequent letter in the text.

The letter returned must be in lower case.

While checking for the most wanted letter, casing does not matter,

so for the purpose of your search, "A" == "a".

Make sure you do not count punctuation symbols, digits and whitespaces, only letters.

If you have two or more letters with the same frequency,

then return the letter which comes first in the latin alphabet.

For example -- "one" contains "o", "n", "e" only once for each, thus we choose "e".

Input: A text for analysis as a string.

Output: The most frequent letter in lower case as a string.

Example:

mostWanted("Hello World!") == "l"

mostWanted("How do you do?") == "o"

mostWanted("One") == "e"

mostWanted("Oops!") == "o"

mostWanted("AAaooo!!!!") == "a"

mostWanted("abe") == "a"

Ex-3: Appoint Vendor = ( give me a ‘name’ to this pattern )

You are given a sample function which executes service logic. When we call that function,

(a) sould log before & after

(b) should calculate total time it took to execute

(c) should catch error if any thrown

But we sould not implement these concerns in service function. Must be individual functions and compose them to achieve our goal.

Ex-4: Music Player = ( give me a ‘name’ to this pattern )

Song **=> {title :string , duration: number}**

Create list of songs, and *getNextSong()* function which can return next song based user sceduling strategy

Strategies

* SDNS => shortest duration next song
* LDNS => longest duration next song
* FINS => first in next song

Condition: getSongNong() must be polymophic, strategy function must be parameter