Programozás I. gyakorló feladatsor

Témakör: Java kollekciók

Ez egy nemhivatalos Prog1 gyakorló feladatsor, amit a gyakorolni vágyó hallgatóknak készítettem. 11 feladatot állítottam össze a kollekciók témaköréből, melyből az első 8 körülbelül lefedi azt a szintet, amit a félév végi nagyzh-ra tudni érdemes. A maradék három feladatból az első egy összetettebb, gondolkodtatóbb Set feladat, a másik kettő pedig két olyan kollekció lehetséges alkalmazását mutatja be, ami nem része a Prog1 anyagnak.

Minden feladathoz vannak tippek, ezek HTML (vagy XML) stílusban `<segítség>` tag-ek között találhatók. Fehér színű szövegként vannak beírva, így ha kijelölitek, kimásoljátok és beillesztitek valahova, el tudjátok olvasni őket. (De azért próbálkozzatok minden feladattal legalább pár percet, mielőtt ezekhez nyúlnátok!)

Továbbá készült mindegyik feladathoz egy mintamegoldás is, ami tele van kommentekkel, melyek remélhetőleg sokat segítenek a zh-ra való készülésben, valamint egy main metódus, ami segítségével egyszerű inputokat használva tesztelni lehet az elkészült függvényeket. Fontos megemlíteni, hogy itt a formai követelmények (függvény neve, pontos paramétertípusok) nincsenek megadva, mint a Bíró esetében, így egyes teszteket lehetséges, hogy némiképp módosítanotok kell, hogy le tudjátok futtatni. A mintamegoldások kommentjei néhol kitérnek arra is, hogy egyes feladatokat hogy lehet máshogyan megoldani, milyen egyéb típusokat lehet használni.

Jó gyakorlást és sikeres készülést mindenkinek, aki arra szánta el magát, hogy ebből a feladatsorból gyakoroljon!

(Kérdésekkel kapcsolatos információk a feladatsor végén vannak.)

1. feladat: a)

Írj egy függvényt, ami nem vár paramétert, és egy olyan kollekcióval tér vissza, ami megvalósít a {0, 1, 2, 3, 4} és a {3, 5, -10, -625, 491075992, 0} halmazok között egy injektív, de nem szürjektív leképezést!

```
@return a leképezés

<segítség tárgy=”dimat1”>
                                                    </segítség>

<segítség>
                                                    </segítség>
```

1. feladat: b)

Írj egy függvényt, ami nem vár paramétert, és egy olyan kollekcióval tér vissza, ami megvalósít a {0, 2, 4, 6, 8} és {"alma", "körte", "szilva", "barack", "narancs"} halmazok között egy bijektív leképezést!

```
@return a leképezés
```

2. feladat

Valaki összekeverte az értékeinket: mindet szöveggé alakította, és beledobálta egy listába. A feladatod, hogy szétválogasd őket típus szerint: igazságérték, egész szám, szöveg. Ha az adott szöveg a "true" vagy "igaz", vagy "false" vagy "hamis", akkor alakítsd át a megfelelő igazságértékké. Ha az adott szöveg egy egész szám, akkor alakítsd át azzá a számmá. Figyelj arra, hogy akár 16 jegyű egészeket is kellhet eltárolni! Amelyik szöveg nem esik bele egyik kategóriába sem, az maradjon szöveg. Az igazságértékek, számok és szövegek listáját rakd bele egy tömbbe, és ezt adja vissza a függvény! A szűrés tartsa meg az azonos kategóriába eső elemek sorrendjét!

```
@param szurnivalo a szűrni kívánt lista
@return a szűrt listák tömbje

<segítség>

                    </segítség>

<segítség>
                            </segítség>
```

3. feladat

Írj egy függvényt, ami egy egész számokból álló listát vár paraméterként. A függvény adjon vissza egy új listát, amelyben az eredeti lista páros elemei szerepelnek megduplázva.

```
@param lista a lista, amelyből egy újat szeretnénk készíteni
@return az új lista
```

4. feladat

Elméleti feladat: hány különböző igazságértékeket tartalmazó halmazt kell létrehozni (és valahogyan feltölteni elemekkel), hogy biztosan legyen köztük kettő egyenlő?

```
<segítség>
        </segítség>
```

5. feladat

A világhírű SlickClips284 legújabb dalai egy Suno nevű mesterséges intelligencia által lettek generálva. Valósítsd meg a generált dalokat reprezentáló osztályt a megadott minta kiegészítésével!

- Készíts el egy kétparaméteres konstruktort, ami várja a dal címét és hosszát, és beállítja a megfelelő adattagokat. A többi adattagot inicializáld értelemszerűen.
- Készíts el egy másik konstruktort, ami a cím és hossz mellett tetszőlegesen sok címkét vár paraméterben. A címkéket tárold el a nekik megfelelő adattagban.
- Készítsd el az *atnevez* metódust, ami egy szöveget vár paraméterben, és átnevezi a dalt a kapott szövegre. (Minek nevezzük az ilyen típusú metódusokat?)
- Készítsd el az *isInstrumentalis* metódust, ami egy igazságértékkel tér vissza, hogy az adott szám instrumentális-e. Akkor mondunk egy számot instrumentálisnak, ha nincs dalszövege.
- Készítsd el a *hasCimke* metódust, ami egy igazságértékkel tér vissza, hogy a paraméterben érkező címkével (szöveg) rendelkezik-e a dal.
- Készítsd el a statikus *general* metódust, ami megvalósítja egy új dal generálását. A metódus várja a dal címét, a címkéket egy tömbként, és a dalszöveget (mely tetszőlegesen sok szövegből állhat). A metódus visszatérési értéke a generált dal. A dal hossza úgy számolódik ki, hogy ha instrumentális, akkor pontosan 90 másodperc hosszú lesz. Ha a címkék bármelyikében szerepel a "rock" kifejezés, akkor a Suno nem bír magával, és legalább 3 perc hosszúra csinálja, különben befejezi a generálást legfeljebb 2 percnél. Továbbá ha a címkék között szerepel pontosan a "pop" elem, akkor még 2 perc hosszúra sem nyúlik ki a szám, hanem annyi másodperccel lesz rövidebb nála, ahány sorból áll a dalszöveg. (Vigyázat, a Suno jobban szereti a rock-ot, mint amennyire nem szereti a pop-ot!) Ha legalább 4 címkéje van a rock számnak, egy "symphonic rock" számról beszélünk, vagy legalább 7 versszakból áll, akkor a Suno egy epikus 6,4 perces számmal rukkol elő.
- Készítsd el a statikus *bovit* metódust, ami egy meglévő dalt bővít egy újabb részlettel. A metódus várja az eredeti dalt, az új címet, az új címkéket egy tömbként, és az új dalszöveget (mely tetszőlegesen sok szövegből állhat). A metódus visszatérési értéke a bővített dal. Az új dal címe az új cím lesz, valamint a címkéi is, a dalszöveg pedig az eredeti dalszöveghez fűződik hozzá. Az új dal hossza a következőképpen alakul a régi dal hosszán felül: ha az új rész instrumentális, akkor pontosan másfél perc új melódiával gazdagodik, különben csak 0,6-tal. Ha a címkék bármelyikében szerepel a "rock" kifejezés, akkor a Suno megduplázza az eredeti hosszát. Ha viszont több új versszakkal bővítjük a dalt, mint amennyi eredetileg volt, akkor helyette háromszorozódik a hossz. Különben, ha az új címkék között mindegyik régi szerepel, és ezen kívül legalább

két új is, akkor a Suno egy bug folytán mindössze egyetlen másodperccel hosszabbítja meg a dalt.

Az összes elkészített metódus legyen publikus láthatóságú!

6. feladat

Írj egy függvényt, ami megszámolja, hogy a paraméterként kapott szöveg-szöveg leképezés kulcsai és értékei között hány különböző 4 betűs szó van!

```
@param map a vizsgálandó leképezés
@return a szavak száma
```

<segítség>

</segítség>

7. feladat

Magdi néni szeretne utánajárni, hogy a 3.A nebulói közül kik buknak meg. Sajnos, bár Magdi néni Javában olvasta be az adatokat, az osztály fogalmát csak a 3.A kapcsán ismeri, ezért különböző kollekciók segítségével tárolta el a tanulók év végi jegyeit. Egy leképezésben a diákok neveihez újabb leképezéseket rendelt, amik a tantárgy nevéhez az érdemjegyet (1-5) rendelték. Írj egy függvényt, ami ilyen formában várja az adatokat, és térjen vissza azoknak a diákoknak (csak a nevüknek) a listájával, akik megbuknak, azaz legalább egy tantárgyból 1-est szereztek! A lista legyen névsor szerint rendezve!

```
@param diakok a diákok és érdemjegyeik
@return a megbukó diákok nevei
```

<segítség>

</segítség>

<segítség téma=”rendezés”>

</segítség>

8. feladat

Péter kedvenc száma a 3, így természetesen megszállottja a 3-mal való szorzásnak. Nem is szereti, amikor egy egész-egész leképezésben, ami nem az övé, az N kulcshoz tartozó érték N*3. Amíg Karcsi nem figyelt, ellopta a leképezését, és most ki akarja törölni azokat a kulcs-érték párokat, amikre a fenti szabály igaz. De sietnie kell!

Valósítsd meg a fenti működést mindössze egyetlen sorban! (1 sor vagy 1 egysoros ciklus lehet benne.)

```
@param lekepezes Karcsi szeretett leképezése (Integer-Integer leképezés)
```

9. feladat

Vivi hónapok óta feljegyzi, ami vele történt, most azonban megelégelte a sok papírt, és össze akarja gyűjteni a feljegyzéseket egyetlen naplóba. Egy feljegyzést egyértelműen beazonosít egy dátum (szöveg), és tartozik még hozzá maga a feljegyzés szövege. Vivi azonban nem akarja beleírni az összes feljegyzést a naplójába. Vannak köztük kínosak és nagyon rövidek, amiket felesleges átírnia. Írd át a feljegyzések listáját a naplóba, de csak olyan feljegyzéseket, amelyeknek a szövege több mint 5 szó hosszú, és nem tartalmazza a "kínos", "szerelmes" és "anime" szavak egyikét sem.

```
@param feljegyzesek a feljegyzések: Map.Entry-k listája
@return a napló, egy asszociatív tömb
```

<segítség>

</segítség>

Vigyázat! Innenről nehezebb, a nagyzh szintjét meghaladó feladatok következnek!

10. feladat

Készítsük el Integer halmazok egy osztályozását egy osztály és egy Map segítségével! Akkor tartozzon két halmaz egy osztályba, ha az elemeik szorzatának előjele ugyanaz. (-1, 0, 1, ez legyen a három osztály neve is.) Készítsd el a HalmazOsztalyozas osztályt, és egy függvényt, ami visszaad egy olyan Map-et, amire a map.get(new HalmazOsztalyozas(halmaz)) hívás visszaadja az osztályt, amelyhez tartozik a halmaz. Ha nem tudod, hogyan kezdj hozzá a feladathoz, olvasd el az 1-es segítséget!

```
@return a halmazok osztályozását megvalósító Map
```

```
<segítség id=”1”>

                                                    </segítség>

<segítség>


                          </segítség>

<segítség tárgy=”dimat1>
                                                  </segítség>
```

Az utolsó két feladatot órán nem tárgyalt kollekciókkal a legegyszerűbb megoldani. Nézz utána, hogy mivel lehet érdemes! Gyakorlásképpen megoldhatod lista-halmaz-leképezés trióval is, de a mintamegoldások nem ezeket a kollekciókat helyezik reflektorfénybe.

11. feladat

Áronkának anyukája mossa ki a ruháját. Minden nap valamennyi pólót kimos, amit utána odarak Áronka szekrényébe. Áronka mindig a legfelső pólót veszi fel, minden reggel egyet. Az anyukája délután mos ruhákat, akkor rakja az újakat a szekrénybe, a korábban kimosott pólók tetejére. Írj egy függvényt, ami tetszőlegesen sok szövegtömböt vár paraméterként. Ezekben a tömbökben az adott napon kimosott pólók színei vannak (olyan sorrendben, ahogy az anyukája bepakolta őket a szekrénybe). A függvény adjon vissza egy tömböt. A tömb i. indexén az i. napon felvett póló színe szerepeljen. Olyan hosszú legyen a tömb, ahány napig lesz pólója Áronkának a paraméterben kapott adatok alapján. Az első, paraméterben kapott tömb lentről felfelé, az alapból szekrényben lévő pólók színeit tartalmazza.

```
@param napok a napok, a kimosott pólók színeivel
@return az Áronka által felvett pólók színei (sorrendben)
```

12. feladat

Jánoska játékokat szeretne Karácsonyra. Sokat. Éppen ezért levelet ír a Jézuskának, arról, hogy milyen játékokat kér. Használd fel az előre elkészített Jatek osztályt, és egészítsd ki a megfelelő módon! Írj egy függvényt, ami várja Jánoska levelét (egy listát a játékokról), és egy egész számot, hogy hány játékot tud megvenni neki a Jézuska. A függvény térjen vissza azon játékok nevének a halmazával, amiket a Jézuska megvásárol. Mivel a Jézuska nagyon aranyos, ezért a legfontosabb N játékot veszi meg Jánoskának. (Egy játék minél fontosabb, annál nagyobb a prioritas adattagjának az értéke.)

```
@param level Jánoska levele
@param n hány játékot tud megvenni Jézuska
@return a megvásárolt játékok
```

<segítség>

</segítség>

A feladatokat igyekeztem legjobb tudásom és az Internet (gyakjegyzetek, javapoint.com, ChatGPT) segítségével megírni, de hibák, pontatlanságok és kevésbé érthető részek maradhattak benne. Ha a fentiek egyikét találod, bátran keress meg egy CooSpace üzenet formájában, Neptun kódom UQ13LD. Igyekszek minden felmerülő kérdésre válaszolni, ha ezzel segíthetek felkészülni a nagyzh-ra.

Goldman Júlia ajánlásával

A feladatokat és mintamegoldásokat készítette: Kozma Kristóf