Korszerű Programozási Technikák
Pót-NagyZH

A feladat egy étrendkiegészítő hatóanyagának vizsgálatával kapcsolatos adatok kezelése.

Adott a **Vizsgalat** osztály, ami minden adatot tárol. A vizsgálat résztvevőit számokkal azonosítjuk (**unsigned**), és mindenkihez az alábbi adatok tartoznak:

- a résztvevő adatai (**Szemely** osztály: vezetéknév, keresztnév, nem, életkor),
- a szer szubjektív értékelése a résztvevő által (**int**, 1-től 10-ig),
- placebót kapott-e az adott résztvevő (**true**) vagy tényleges hatóanyagot (**false**).

Mindhárom adathoz van egy-egy asszociatív tároló, ahol a kulcs mindig a résztvevő azonosítója. Ezen három adattagot kell használni a feladatokban. Tipp: az asszociatív tárolók értékeinek elérésére konstans kontextusban az *at* függvényt, nem konstans kontextusban a `[]` operátort lehet használni. Minden feladatban feltételezhetjük, hogy a **Vizsgalat** „konzisztens" adatokkal lesz tesztelve, vagyis, ha egy azonosító szerepel valamelyikben kulcsként, akkor a másik kettőben is ott lesz.

A *kiir* függvény már adott, a tesztkód több helyen használja is: adott kimeneti folyamra írja ki a vizsgálat tartalmát azonosítókkal vagy azok nélkül.

A **kimenet-minta.txt** fájlban megtekinthető a példa kimenet.

Alap feladatok (7 pont)

Írjuk meg a **Vizsgalat** osztályba az alábbi függvényeket.

- Legyen egy *setEredmeny* függvény, amellyel egy új résztvevőt és a szubjektív értékelését tudjuk megadni. Az első paraméter az azonosító szám (**unsigned**), a második a résztvevő adatai (**Szemely**), a harmadik pedig az értékelés (**int**). **(2 pont)**
- Legyen egy *setPlaceboLista* függvény, amellyel feltölthetők a placebóval kapcsolatos adatok. A két paraméter: egy lista azonosítókkal, és egy logikai érték, hogy placebóról van-e szó. Az összes felsorolt azonosítóhoz az adott logikai értéket állítsuk be. **(2 pont)**
- Legyen egy *kiirAdottErtekeles* függvény, amelyik kap paraméterként egy értékelést (**int**), és kiír két számot: hogy hány placebós és hány hatóanyagos résztvevőhöz van bejegyezve a megadott értékelés. A formátum nem lényeges. **(3 pont)**

Beolvasó adapter (23 pont)

Írjunk egy adapter osztályt, amelyik képes beolvasni egy **Vizsgalat** tartalmát több különböző fájlból.

A feladat megoldásához adott a *tombOlvaso* sablon függvény (**tombolvaso.h**), ami adott **T** sablonparaméter típusú, azonosító számokhoz rendelt adatokat tud beolvasni egy konkrét fájlból. A függvény első paramétere a fájl neve, a második egy vektor, ami azonosító és **T** párokból áll. A függvény a vektor végére szúrja be a beolvasott adatokat, egyenként.

A fájlok az alábbiak (összesen 7 db):

- **resztvevok-1.txt**, **resztvevok-2.txt**, amik a résztvevők személyes adataikat tárolják (**Szemely**).
- **ertekelesek-1.txt**, **ertekelesek-2.txt**, **ertekelesek-3.txt**, amik a résztvevőkhöz rendelt értékeléseket tárolják (**int**).
- **placebok-1.txt**, **placebok-2.txt**, amik a résztvevőkhöz tárolják, hogy placebóról van-e szó (**bool**).

A fájlok a *main* függvény elején generálódnak a futtatás aktuális könyvtárába (QtCreator-ban a build könyvtár), ezután megtekinthetőek, ha kell, teendő nincs velük.

Tehát például a **tombOlvaso<bool>("placebok-1.txt",vec);** hívással olvashatjuk be a **placebok-1.txt** fájl tartalmát egy **vec** vektorba, amely **vector<pair<unsigned,bool>>** típusú kell, hogy legyen, és ekkor minden pár egy azonosító, és a hozzá tartozó logikai érték, hogy az adott résztvevő placebót kapott-e.

A *tombOlvaso* függvényt csak egyszer, a **mutex**-es feladatban, a megadott módon és mértékben szabad módosítani.

Magukról az adatokról azt tudhatjuk biztosan, hogy konzisztensek, vagyis minden azonosító pontosan háromszor fordul elő a hét fájlban: egyszer a két placebós, egyszer a három értékeléses, és egyszer a két résztvevős fájl valamelyikében.

- Válassz egyet az **Interfesz1**, **Interfesz2** vagy **Interfesz3** osztályok közül, és abból származzon az elkészülő adapter, **Adapter1**, **Adapter2** vagy **Adapter3** néven. Mindegyiknek a *beolvas* tisztán virtuális függvénye végzi a beolvasást. Az interfészek csak a beolvasandó **Vizsgalat** objektum átadási módjában különböznek, és a kódjukat nem szabad módosítani. **(3 pont)**

Írd meg az adapter *beolvas* függvényét, az alábbi útmutatások szerint.

- Legyen három megfelelő vektor, ami **unsigned** és **T** párokat tárol, ahol **T** az alábbiak közül kerül ki: **Szemely**, **int**, **bool**. **(2 pont)**
- Hívd meg a *tombOlvaso* függvényt minden fájlra pontosan egyszer, beolvasva ezáltal az adatokat az említett három vektorba. **(2 pont)**
- A hét függvényhívás fusson hét külön szálon, párhuzamosan. **(3 pont)**
- Az adatverseny kivédése végett bővítsd a *tombOlvaso* függvényt egy plusz **mutex** paraméterrel, amely a vektorba való beszúrást védi (csak azt, a párhuzamosítást ne tegyük tönkre). Az adapterben használjuk is ezt az új paramétert a függvény meghívásakor. A plusz pont azért jár, ha a különböző vektorok beolvasása egymással sem interferál: vagyis az egyik vektorba való beszúrás csak az ugyanabba való, másik szálról történő másik beszúrást zárja ki, a másik vektorhoz való hozzáférést nem. Tipp: Ha nem megy ez a feladat, akkor a későbbi tesztelés végett inkább a szálak is egymás után fussanak, mert **vector** esetén az adatverseny katasztrofális lehet. **(2+1 pont)**

Miután megtörtént a tényleges beolvasás, a *beolvas* függvényben még fel kell tölteni a három vektorban eltárolt adatokkal a **Vizsgalat** objektumot. Mivel a pár-vektorok nem a legjobb input formátum erre a célra, itt most szabad a rendundáns adattárolás, ideiglenes tárolók létrehozása. Emlékeztetőül: minden azonosító mindhárom vektorban pontosan egyszer szerepel.

- Töltsd fel a résztvevőket és értékeléseiket a **Vizsgalat** osztály *setEredmeny* függvényével.**(3 pont)**
- Töltsd fel a placebo-adatokat a **Vizsgalat** osztály *setPlaceboLista* függvényével. **(3 pont)**

A **Vizsgalat** tartalmát a tesztkód a **beolvasott.txt** fájlba menti el. A **beolvasott-minta.txt** fájlban megtekinthető, hogyan kellene kinéznie a vizsgálatnak a beolvasás után.

- A beolvas függvény írja ki, hogy a legelejétől a végéig hány másodperc telt tel. **(2 pont)**
- Írd meg valamelyik másik interfészhez is az adaptert, de úgy, hogy csak az imént megírt adaptert és beolvasó függvényét használod fel. Rendundáns adattárolással nem kell foglalkozni. **(2 pont)**

Választható feladat (10 pont)

Válassz ki **egyet** az **A**, **B** vagy **C** sorozatból. Mindegyik 10 pontot ér, és a legjobb előrehaladás számít.

(A) Lekérdezés

Legyen a **Vizsgalat** osztályban egy *lekerdezes* függvény, ami átlagos értékelést számol bizonyos szűrőfeltételekkel megszabott esetekre. Az előző feladatoktól függetlenül is teljes értékűen megoldható, de a tesztelő kód és a példakimenet az adapter által beolvasott **Vizsgalat** objektumból indul ki.

- A paraméterek szűrőfeltételek. Az első paraméter egy logikai érték, hogy placebós (**true**) vagy hatóanyagos (**false**) esetekre vonatkozzon a lekérdezés. **(2 pont)**
- A második paraméter egy alsó korhatár, a harmadik egy felső korhatár, mindkettő egész szám. Ha bármelyik nulla, az azt jelenti, hogy az adott korhatár nincs megadva, vagyis adott irányban nem szűrünk. A két korhatár közé eső esetekre vonatkozzon a lekérdezés, a határokat is beleértve. **(2 pont)**
- A negyedik paraméter egy egész szám, ami a résztvevő nemét jelöli, 1=csak nők, 2=csak férfiak, 3=mindkét nem körében történjen a lekérdezés. **(2 pont)**
- A visszatérési érték a minden szűrőfeltételnek eleget tevő esetek körében az értékelések átlaga (**double**). **(2 pont)**
- Amennyiben a szűrőfeltételeknek egyetlen eset sem felel meg, dobjon a függvény egy saját, belső osztály típusú kivételt (**std::exception** gyerekosztály, tetszőleges üzenettel). **(2 pont)**

(B) Megfigyelők

Lehessen kétféle megfigyelőt rendelni a **Vizsgalat** osztályhoz. A *main*-ben lévő tesztelő kód kötött, minden más szabadon megválasztható: pontosan mikor történik értesítés, hogyan tárolódnak a megfigyelők, milyen legyen az értesítő függvény, milyen adattagok legyenek a megfigyelőkben stb. Még közös, absztrakt megfigyelő interfész sem feltétlen kell.

- Lehessen megfigyelőket regisztrálni a **Vizsgalat** osztályba egy *regisztral* függvénnyel. Ennek paraméterként vagy egy **MegfigyelEredmeny**, vagy egy **MegfigyelPlacebo** objektumot adhatunk át, pointerként. (Lehet polimorfizmust vagy függvénytúlterhelést is használni.) **(3 pont)**
- A **MegfigyelEredmeny** konstruktora paraméterben egy **promise<unsigned>** objektumot kap, pointerként, valamint egy vezetéknevet, és egy keresztnevet. Amint a megfigyelt **Vizsgalat** objektumhoz a *setEredmeny* függvény az adott nevű személyt adná hozzá, a **promise<unsigned>**

objektum állítsa be a küldendő üzenetet az adott személy azonosító számára. Feltételezhetjük, hogy az adott név pontosan egyszer bukkan majd fel. **(4 pont)**

- A **MegfigyelPlacebo** konstruktora paraméterben egy azonosító számot kap. Amikor a megfigyelt **Vizsgalat** objektumba a *setPlaceboLista* az adott azonosítót szúrná be, írjuk ki egy sorban az azonosító számot, és hogy placebós volt-e az eset, vagy sem. A formátum nem fontos. **(3 pont)**

(C) Szüneteltetés

Legyen egy *getResztvevoSzam* függvény a **Vizsgalat** osztályban, ami életkor szerint csoportosítva adja vissza a résztvevők számát – ennek a függvénynek az elérése azonban legyen szüneteltethető. Mindezt felesleges késleltetés, busy-waiting, undefined behavior, stb. nélkül kell megoldani, gondolva arra, hogy a függvényhívások egyszerre, több különböző szálról is érkezhetnek.

- A visszatérési érték a feladatnak megfelelően egy **map<int,unsigned>** objektum. A kulcs most az életkor, az érték pedig az adott életkorú résztvevők száma. A függvény a többi feladat érdekében kifejezetten ne legyen const. **(3 pont)**
- A szüneteltetés azt jelenti, hogy legyen egy plusz adattag a **Vizsgalat** osztályban, ami jelzi, hogy most szüneteltetve van-e a *getResztvevoLista* szolgáltatás, vagy sem. Ez kezdetben **false**, vagyis nincs szünet. A *setPiros* függvénnyel legyen beállítható, a *setZold* függvénnyel pedig feloldható a szünet. Amennyiben a *getResztvevoSzam* függvény elején szünet van éppen, addig várakozzon, amíg ez fel nem oldódik (ő maga csak figyelheti a szünetet, nem módosíthatja). Ha a *getResztvevoSzam* függvény elején nincs szünet, illetve amint az feloldódik a *setZold* által, a függvény onnantól szabadon végigfuthat. **(3 pont)**
- Amennyiben már futó *getResztvevoSzam* hívások vannak, amikor a *setPiros* szüneteltetni akar, akkor maga a *setPiros* függvény is várja meg, amíg azok befejeződnek. Amíg a *setPiros* vár, addig új hívások már ne induljanak el. Ennek a feladatnak a helyességét a tesztkód nem vizsgálja. **(4 pont)**