BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2019 július 1
Informatika - magyar nyelv

2. VÁLTOZAT

Megjegyzések:
- **Mindegyik tétel kötelező; a tételeknél teljes megoldásokat kérünk.**
- **A dolgozat minimális átmenő jegye az ötös (5,00).**
- **Munkaidő: három (3) óra.**

<u>Operációs rendszerek</u> TÉTEL

1 Feltételezve, hogy az alábbi C függvények minden utasítása helyesen lefut, válaszoljunk az alábbi kérdésekre:

```
1   void f1(){
2     int i;
3     for(i=0; i<3; i++) {
4       if(fork() == 0) {}
5       wait(0);
6     }
7   }
8   void f2(){
9     int i, p = 0;
10    for(i=0; i<3; i++) {
11      if(p == 0) {
12        p = fork();
13      }
14      wait(0);
15    }
16  }
```

a) Rajzoljuk le az `f1` függvény szülő-folyamatból való meghívása során létrehozott folyamat-hierarchiát.

b) Rajzoljuk le az `f2` függvény szülő-folyamatból való meghívása során létrehozott folyamat-hierarchiát.

c) Írjuk át az `f1` függvényt úgy, hogy ugyanannyi folyamatot hozzon létre, mint az `f2` függvény, de más legyen a hierarchia.
Magyarázzuk meg és rajzoljuk le a folyamat-hierarchiát.

d) Magyarázzuk meg a `wait` rendszerhívás szerepét!

2 Válaszoljunk az alábbi UNIX shell-szkripthez kapcsolódó kérdésekre, feltételezve, hogy egy, a mellékelt szerkezetű katalógusból futtatjuk a következő paraméterekkel: `a d1 alma b d2 korte`.

```
1   #!/bin/bash
2   while [ -n "$1" ]; do
3     if [ -d $1 ]; then
4       for F in `find $1 -type f`; do
5         if file $F | grep -q -v text; then
6           echo "Hiba: $F"
7           continue
8         fi
9         sed "s/^$2//" $F > $F.text
10      done
11      shift
12      shift
13    else
14      shift
15    fi
16  done
```

```
.
├─ d1        - katalógus
│  ├─ a      - szöveges file
│  ├─ b      - szöveges file
│  └─ x.txt  - .txt-re átnevezett PDF file
└─ d2        - katalógus
   └─ b      - szöveges file
```

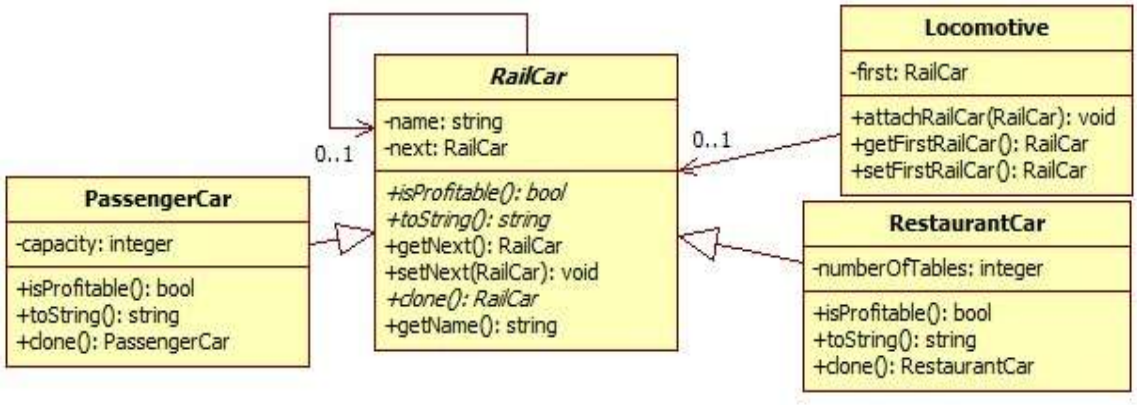
a) Mit ír a képernyőre a szkript végrehajtáskor, és mi a hatása a fenti katalógusszerkezetre, illetve állományokra?
b) Magyarázzuk el részletesen az 5. sort.
c) Hogyan változik a végrehajtás eredménye, ha a 9. sorban az idézőjeleket aposztrófra cseréljük?
d) Magyarázzuk meg**,** mi történik, ha töröljük a 13. és 14. sorokat!

2. VÁLTOZAT

Algoritmusok és Programozás TÉTEL

Írjunk egy programot a C++, Java, C#, vagy Python programnyelvek egyikében az alábbi követelmények szerint:

a) Definiálja a **RailCar** (Vagon), **PassengerCar** (Személykocsi), **RestaurantCar** (Étkezőkocsi), **Locomotive** (Mozdony, mely a hozzá tartozó vagonokat tárolja) osztályokat az alábbi UML-diagram alapján (a diagramban a konstruktorokat nem jelöltük). Az ábrán definiált metódusok közül csak a b) pontban jelölteket kell implementálni; a többit csupán definiáljuk.



- A **RailCar** osztály *name* (vagon neve) attribútuma legkevesebb 3 hosszúságú; a **PassengerCar** *capacity* (férőhelyek) és a **RestaurantCar** *numberOfTables* (asztalok száma) attribútumok szigorúan pozitívak. A megszorításokat a konstruktorok implementálják.
- A **RailCar** absztrakt osztálynak három absztrakt metódusa van: **isProfitable**, **toString** és **clone**. A **PassengerCar** és a **RestaurantCar** osztályok **clone** metódusai az objektum egy másolatát térítik vissza.
- A **Locomotive** osztály **attachRailCar** metódusa a paraméterként átadott vagont adja a már létező vagon-listához úgy, hogy a vagonok a **toString** metódus kimenete szerint alfabetikusan rendezve vannak.
- Egy személyvagon jövedelmező (**isProfitable**), ha férőhelyeinek száma nagyobb, mint 40. Egy étkezőkocsi jövedelmező, ha az asztalok száma (*numberOfTables*) nagyobb, vagy egyenlő mint 20.
- A személyvagonok esetében a **toString** metódus a vagon nevének (*name*) és férőhelyeinek (*capacity) az* összefűzését téríti vissza. Az étkezőkocsik esetén a **toString** metódus a vagon nevének (*name*) és az asztalok számának (*numberOfTables*) az összefűzését téríti vissza.

b) Implementáljuk a következő metódusokat az a) pont UML-ábrájából: a **RailCar, PassengerCar**, **RestaurantCar** osztályok konstruktorait; a **PassengerCar** és a **RestaurantCar** osztályok **toString** metódusát; a **Locomotive** osztály **attachRailCar** metódusát; a **PassengerCar** és **RestaurantCar isProfitable** metódusait; a **PassengerCar clone** metódusát, valamint a **RailCar** osztály **setNext** metódusát.

c) Definiál egy függvényt, melynek egy l **Locomotive** típusú paramétere van. Az l-ből kitörli az összes nem jövedelmező vagont.

d) Definiál egy függvényt, melynek egy l **Locomotive** típusú paramétere van és visszatérít egy **Locomotive** típusú objektumot, ahol a vagonok fordított sorrendben helyezkednek el.

e) Létrehoz a program fő-függvényében (main) egy l **Locomotive** típusú objektumot, melyhez a következő vagonokat adja: három **PassengerCar** típusú objektumot 50, 32 és 40 férőhellyel, illetve két **RestaurantCar** típusú objektumot 25 és 20 asztallal (a nem specifikált mezőknek adjatok tetszőleges értéket). Töröljük ki a listából a nem jövedelmező vagonokat a c) pont használatával, majd hívjuk meg a d) pont függvényét egy l' **Locomotive** típusú objektum létrehozására. Végül jelenítsük meg az l' lista vagonjait.

f) Írjuk meg a **Locomotive** osztály **attachRailCar** metódusához tartozó specifikációt.

- **Specifikáljuk a használt programozási nyelvet!**
- **Ne használjunk a specifikáción kívüli metódusokat (a konstruktorok kivételével)!**
- **Nem szabad rendezett konténereket és előredefiniált rendező műveleteket használni!**

*Az **adattípusokra** használhatunk létező könyvtári függvényeket (Python, C++, Java, C#).*

2. VÁLTOZAT

Adatbázisok TÉTEL

Tekintsünk egy adatbázist, mely egy kórház pácienseiről és kezeléseiről tartalmaz információkat. Feltételezzük, hogy a kórházban nincs két azonos vezeték- és keresztnevű orvos. Az adatbázis szerkezete a következő:

- ***Páciensek*** tábla, attribútumai: **PKód**, **Vezetéknév**, **Keresztnév, Életkor, SzületésiDátum**;
- ***Kezelések*** tábla, attribútumai: **KezKód**, **Név, Leírás, Ár**;
- ***PáciensekKórlapjai*** tábla, attribútumai: **PáciensKód**, **KezelésKód**, **Dátum, OrvosVezetéknév**, **OrvosKeresztnév, OrvosSzakosztály, OrvosRégiség**.

1. Határozzuk meg a fenti táblák elsődleges kulcsait és külső kulcsait!

2. Adjunk meg legalább 3 funkcionális függőséget, melyek NEM kód attribútumra vonatkoznak!

3. Az alábbi szerkezeti módosítások közül melyek szükségesek ahhoz, hogy az adatbázis harmadik normálformában **(3NF)** legyen. Indokoljuk meg választásainkat!

- **a.** Külön tábla létrehozása az orvosok tárolására.
- **b.** A ***Páciensek*** táblában az alábbi megszorítás megadása: **Életkor = aktuális dátum – SzületésiDátum.**
- **c.** Az ***Életkor*** mező törlése a ***Páciensek*** táblából.
- **d.** Külön tábla létrehozása a páciensek életkorának tárolására.
- **e.** A ***PáciensekKórlapjai*** táblában a **Dátum < aktuális dátum** megszorítás létrehozása.

4. Írjunk egy SQL lekérdezést, az eredeti szerkezetre vonatkozóan, mely ekvivalens az alábbi lekérdezéssel:

$$\Pi_{\text{Vezetéknév,Keresztnév}} (\sigma_{\text{OrvosSzakosztály='radiológia'}} (\textbf{\textit{PáciensekKórlapjai}}) \bowtie_{\text{PáciensKód=PKód}} \textbf{\textit{Páciensek}})$$

$$\Pi_{\text{Vezetéknév,Keresztnév}} (\sigma_{\text{OrvosSzakosztály='kardiológia'}} (\textbf{\textit{PáciensekKórlapjai}}) \bowtie_{\text{PáciensKód=PKód}} \textbf{\textit{Páciensek}})$$

5. Írjunk egy SQL lekérdezést, az eredeti szerkezetre vonatkozóan, mely megadja minden olyan szakosztály esetén, mely a legtöbb pénzt hozta a kórháznak a kezelésekből, a szakosztály nevét, az ott végzett kezelések összszámát és az ezekből befolyt pénzösszeget (***Szakosztály, KezelésekSzáma, Összeg***). Minden szakosztályt adjunk meg, mely teljesíti a megadott feltételt!

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p
Definirea clasei abstracte **RailCar** – 1 p din care
atribut – 0.2
constructor si metode **-** 0.8
Definirea clasei **PassengerCar**– 1.1 p din care
relația de moștenire – 0.2
atribut – 0.1
constructor si metode – 0.8
Definirea clasei **RestaurantCar** – 1.1 p din care
relația de moștenire – 0.2
atribut – 0.1
constructor si metode – 0.8
Definirea clasei **Locomotive** – 1.3p din care
atribute– 0.1
constructor - 0.1
metoda attachRailCar - 1.1p
Funcția de la punctul c) – 1.8p din care
signatura corectă - 0.1p
eliminare vagoane neprofitabile- 1.7p
Funcția de la punctul d) – 1.7p din care
signatura corectă - 0.1p
creare lista vagoane - 1.6p
Funcția principală e) – 0.5p
f) Specificaţia metodei **attachRailCar** din clasa **Locomotive**– 0.5p

Notă. Datorită faptului că signatura metodei setFirstRailCar() este incompletă pe diagrama UML, se va acorda punctaj maxim (1.7p) pentru punctul d).

Subiect Baze de date
1. 0.5p (chei primare) + 0.5p (chei externe) = 1p
2. 1p
3. a, c
2 x (0.5p răspuns + 0.5p justificare) = 2p
4. rezolvarea completă a interogării = 2p
5. rezolvarea completă a interogării = 3p
1p of

Subiect Sisteme de operare

Oficiu – 1p

1.a Diagramă cu proces părinte, fii, nepoți și strănepoți, totalizând 8 procese – 1p
1.b Diagramă liniară cu 4 procese: părinte, fiu, nepot, strănepot – 1.5p
1.c Implementare corectă - 0.5p
Explicație – 0.5p
Diagramă – 0.5p
1.d Așteaptă terminarea unui proces fiu – 1p

2.a Afișează eroare pentru x.txt și creează trei fișiere cu extensia .text din care se șterg cuvintele date – 1.5p
2.b Comenzile file și grep – 0.5p
Argumentele –q și –v – 0.5p
Expresia regulară – 0.5p
2.c Caută literalmente $2, fără a substitui valoarea argumentului – 0.5p
2.d Ciclu infinit – 0.5p