BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA KAR

Licenszvizsga, 2024 szeptember 3.
Informatika - magyar nyelv

2. VÁLTOZAT

MEGJEGYZÉSEK

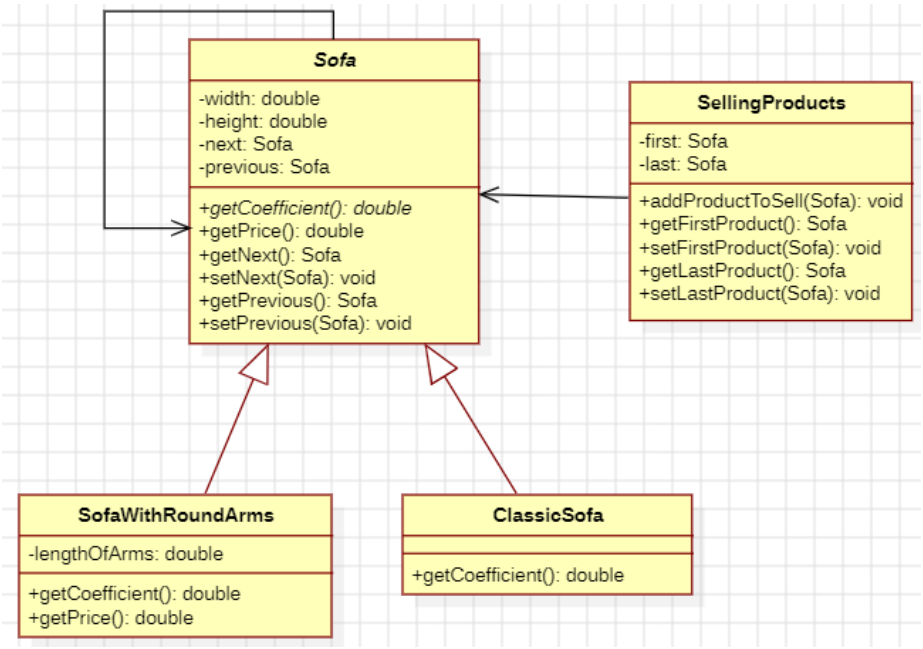
- Mindegyik tétel kötelező; a tételeknél teljes megoldásokat kérünk.
- A dolgozat minimális átmenő jegye az ötös (5.00).
- Munkaidő: három (3) óra.

Algoritmusok és programozás TÉTEL

- **A használt programozási nyelvet meg kell jelölni.**
- **A megírandó programokban nem szükséges a dinamikusan lefoglalt memória felszabadítása.**
- **A megfelelő programozási stílus hiánya (azonosítók beszédes elnevezése, indentálás, megjegyzések ha szükségesek, a kód olvashatósága) az adott tételhez tartozó pontszám 10%-nak az elvesztéséhez vezet.**
- **Tilos új adattagok és metódusok hozzáadása a kijelentésben említetteken kívül, leszámítva a konstruktorokat. Tilos a kijelentésben megadott adattagok és metódusok láthatóságának a megváltoztatása.**
- **Előre definiált rendezett konténerek és rendezési valamint keresési műveletek használata tilos.**

Adattípusokhoz *használhatóak a megfelelő programozási nyelvek* (*C++, Java, C#*) *létező könyvtárai.*

a) Adott az alábbi UML diagram, mely a ***Sofa*** (Kanapé), **SofaWithRoundArms** (KanapéKerekKarokkal), **ClassicSofa** (KlasszikusKanapé) és **SellingProducts** (EladóTermékek, az eladó kanapékat tárolja) osztályokat tartalmazza. A konstruktorok nem jelennek meg a diagramon.



- A ***Sofa*** osztály *width* és *height* adattagjai (szélesség és magasság) és a **SofaWithRoundArms** osztály *lengthOfArms* adattagja (a karok méterben mért hossza) szigorúan pozitívak kell legyenek. A konstruktorok meg kell bizonyosodjanak a megkötések betartásáról.

- A ***Sofa*** absztrakt osztálynak van egy *getCoefficient()* absztrakt metódusa, amely visszatérít egy együtthatót, amelyet a kanapé árának kiszámításánál alkalmazunk.
- Egy **ClassicSofa** típusú objektum árának a kiszámításához 1.5-ös együtthatót használunk, míg egy **SofaWithRoundArms** típusú objektum árának esetén az együttható 2.
- Egy kanapé árát úgy számoljuk ki, hogy megszorozzuk az együtthatót a *width* és *height* összegével. **SofaWithRoundArms** esetén, ehhez az összeghez hozzáadódik a karok ára. Egy méter kar ára 10 lej.
- A **SellingProducts** osztály *addProductToSell(Sofa)* metódusa hozzáadja a paraméterként megadott objektumot a már létező kanapék sorozatához. A hozzáadás olyan módon kell történjen, hogy a kanapék ár szerinti csökkenő sorrendben való kiíratása lineáris időt vegyen igénybe.

Írjatok programot a C++, Java, vagy C# nyelvek valamelyikében, a következő kérésekkel:

a1) Deklaráljátok az összes osztályt, adattagot és metódust a fenti diagramnak megfelelően.

Csak az alábbi metódusokat implementáljátok:

a2) A **SofaWithRoundArms** osztály *getPrice()* metódusát**.**
a3) A **Sofa, SofaWithRoundArms** és **SellingProducts** osztályok konstruktorát.
a4) A **SellingProducts** osztály *addProductToSell(Sofa)* metódusát.
a5) A **ClassicSofa** osztály *getCoefficient()* metódusát és a ***Sofa*** osztály *setPrevious(Sofa)* metódusát.

b) Definiáljatok egy függvényt, amely paraméterként egy **SellingProducts** típusú *s* objektumot kap és lineáris időben kiíratja az *s*-ben található kanapék árát növekvő sorrendben. Azon megoldások, melyek nem tartják be az időbonyolultságot, részpontszámot érnek.
c) Definiáljatok egy függvényt, amely paraméterként egy **SellingProducts** típusú *s* objektumot, valamint két *startPrice* és *endPrice* valós értéket kap és kitörli az *s*-ből azokat a kanapékat, melyek ára a [*startPrice, endPrice*] intervallumban található. Az időbonyolultság $O(n)$ kell legyen (n a **SellingProducts** objektumhoz tartozó listában szereplő kanapék száma). Azon megoldások, melyek nem tartják be az időbonyolultságot, részpontszámot érnek.
d) Építsetek fel egy **SellingProducts** típusú objektumot, amelyhez adjatok hozzá 4 kanapét (a meg nem határozott adattagoknak adjatok tetszőleges értékeket): két **ClassicSofa** típusút és két **SofaWithRoundArms** típusút 1.2 illetve 0.9 méter hosszúságú karokkal. Hívjátok meg a b) majd a c) alpontnál szereplő függvényt (válasszatok tetszőleges *startPrice* és *endPrice* értékeket).
e) Határozzátok meg a **SellingProducts** osztály *addProductToSell(Sofa)* metódusának időbonyolultságát a *legjobb*, *átlag* és *legrosszabb* esetben.

Adatbázisok TÉTEL

1. Feladat (4 pont)

Egy kagylógyűjtő verseny szervezői az alábbi információkat tárolják egy relációs adatbázisban:

- Egy versenyzőről tároljuk: versenyző kódját, nevét, e-mail címét, származási országát.
- Egy partszakaszról tároljuk: partszakasz kódját, nevét, leírását, helyszínét.
- Egy verseny esetén tároljuk: verseny kódját, megnevezését, verseny dátumát, kezdés- és befejezés időpontját (mindkettőt órában adjuk meg), valamint a partszakaszt, ahol a verseny megszervezésre került. Egy adott partszakaszon több kagylógyűjtő versenyt is szervezhetnek.
- Egy kagylótípusról tároljuk: kagylótípus kódját, megnevezését, leírását.
- Minden verseny esetén a versenyzők különböző típusú kagylókat gyűjtenek azon a partszakaszon, amelyen az adott versenyt megrendezték. Az adatbázisban csak azt tároljuk el, hogy mely versenyző mely versenyen mely kagylótípusból hány darabot gyűjtött össze (például, az *r* versenyző: a *v1* versenyen 10 db-ot gyűjtött össze a *k1* típusú kagylóból, 15 db-ot a *k2* típusúból; míg a *v2* versenyen 50 db-ot gyűjtött a *k1* típusúból, 15 db-ot a *k2* típusúból, míg 20 db-ot a *k3* típusú kagylóból stb.).

Tervezzük meg a fenti információknak megfelelő relációs adatbázis sémáját, melynek táblái BCNF-ben vannak. Jelöljük az elsődleges- és külső kulcsokat, valamint a relációk további kulcsjelöltjeit. A relációk sémáját az alábbiakban megadott módok egyikének megfelelően adjuk meg:

* példa a Diakok, Kurzusok és Vizsgak táblákra vonatkozóan:

V1. Adatbázis-diagram a táblák megadásával: az elsődleges kulcsok egyenes vonallal, míg a tábla további kulcsjelöltjei szaggatott vonallal vannak aláhúzva; a fiú táblában megadott külső kulcsot az apa táblában szereplő elsődleges kulccsal/kulcsjelölttel egy direkt él köti össze (lsd. a Vizsgak tábla DiakKod attribútuma és a Diakok tábla DiakKod attribútuma közötti él).

V2.

Diakok[DiakKod, CNP, Vezeteknev, Keresztnev]
Kurzusok[KurzusKod, Megnevezes, Kreditszam]
Vizsgak[DiakKod, KurzusKod, Jegy]
Az elsődleges kulcsok alá vannak húzva egyenes vonallal, míg a további kulcsjelöltek szaggatott vonallal. {DiakKod} külső kulcs a Vizsgak relációban, mellyel a Diakok tábla {DiakKod} attribútumára hivatkozunk. {KurzusKod} külső kulcs a Vizsgak relációban, mellyel a Kurzusok tábla {KurzusKod} attribútumára hivatkozunk.

2. Feladat (5 pont)

Tekintsük az alábbi relációsémákat egy magánóvoda részleges adatbázisára vonatkozóan:

Csoportok[CsoportKod, CsoportNev, Teremszam]
Szulok[SzuloKod, SzuloNev, Cim, Telefonszam]
Gyerekek[GyerekKod, GyerekNev, *SzuloKod*, SzuletesiDatum, *CsoportKod*]
Dijbefizetesek[BefizetesKod, *GyerekKod*, BefizetesiDatum, BefizetettOsszeg]

Az elsődleges kulcsok alá vannak húzva, míg a külső kulcsok dőlt betűvel vannak szedve, nevük pedig megegyezik a hivatkozott tábla elsődleges kulcsának nevével.

a. Írjunk egy SQL lekérdezést, mely minden olyan csoport esetén, amelyben történt legalább egy díjbefizetés, megadja a csoport kódját, nevét és a csoportba járó gyerekekhez tartozó díjbefizetések (BefizetettOsszeg) összértékét!

b. Tekintsük a Csoportok, Szulok és Gyerekek relációk alábbi előfordulásait:

Csoportok:

Csoport Kod	Csoport Nev	Terem szam
1	Katicák	1
2	Mókusok	5
3	Méhecskék	8

Szulok:

Szulo Kod	Szulo Nev	Cim	Telefonszam
1	P1	A1	1111111111
2	P2	A2	2222222222
3	P3	A3	3333333333

Gyerekek:

Gyerek Kod	Gyerek Nev	Szulo Kod	Szuletesi Datum	Csoport Kod
1	C1	1	2018.02.12	1
2	C2	2	2018.06.25	1
3	C3	2	2018.06.25	1
4	C4	3	2018.06.30	1
5	C5	1	2020.04.15	2
6	C6	1	2022.06.05	3

b1. A fentebb megadott reláció előfordulások esetén adjuk meg az alábbi lekérdezés kiértékelésének eredményét. Adjuk meg az oszlop(ok) nevét és értékét. A választ NEM kell megindokolni!

```
SELECT sz.SzuloNev, sz.Cim
FROM Szulok sz
    INNER JOIN Gyerekek gy ON sz.SzuloKod = gy.SzuloKod
WHERE gy.CsoportKod = (SELECT cs.CsoportKod
                       FROM Csoportok cs
                       WHERE cs.CsoportNev = 'Katicák')
INTERSECT
SELECT sz.SzuloNev, sz.Cim
FROM Szulok sz
    INNER JOIN Gyerekek gy ON sz.SzuloKod = gy.SzuloKod
GROUP BY sz.SzuloKod, sz.SzuloNev, sz.Cim
HAVING COUNT(DISTINCT gy.CsoportKod) = 1
```

b2. Döntsük el, hogy a Gyerekek reláció fentebb megadott előfordulása kielégíti-e az alábbi funkcionális függőségeket vagy sem! Indokoljuk meg a választ!

- {GyerekKod} → {SzuloKod}
- {SzuloKod} → {CsoportKod}.

Operációs rendszerek TÉTEL

1. Feladat (5 pont) Válaszoljunk a következő kérdésekre, melyek az alábbi kódrészletből fordított `./a.out` program végrehajtására vonatkoznak, tudva, hogy minden szükséges fejlécállomány jelen van, a `fork` és az `execlp` rendszerhívások végrehajtása sikeres, továbbá az `fa` és `fb` FIFO állományokat létrehoztuk előzőleg. A `bash` parancs a `-c` opcióval meghívva végrehajtja a `-c` opciónak értékül adott karaktersorban található parancsot.

```
 1  int main(int argc, char** argv) {
 2    int fa, fb, k; char s[32];
 3
 4    if(fork() == 0) {
 5      execlp("bash","bash","-c","sort <fa >>fb",NULL);
 6      exit(0);
 7    }
 8    if(fork() == 0) {
 9      fa = open("fa", O_WRONLY);
10      for(k=1; k<argc; k++) {
11        write(fa, argv[k], strlen(argv[k]));
12        write(fa, "\n", 1);
13      }
14      close(fa);
15      exit(0);
16    }
17    if(fork() == 0) {
18      fb = open("fb", O_RDONLY);
19      while((k = read(fb, s, 32)) > 0) {
20        write(1, s, k);
21      }
22      close(fb);
23      exit(0);
24    }
25    wait(NULL); wait(NULL); wait(NULL);
26    return 0;
27  }
```

a) Mit fog a képernyőre írni az alábbi végrehajtás? Indokoljuk a választ.

```
./a.out a d b c
```

b) Hogyan befolyásolja a végrehajtást, ha a 8-16 sorokban található IF blokkot áthelyezzük a 3. és 4. sor közé? Indokoljuk a választ.

c) Mit fog a képernyőre írni az alábbi végrehajtás, ha a 20. sorban található `1` számjegyet `2`-re módosítjuk? Indokoljuk a választ.

```
./a.out y z x > /dev/null
```

d) Hogyan befolyásolja a végrehajtást, ha a 14. sort kikommentezzük? Indokoljuk a választ.

e) Hogyan befolyásolja a végrehajtást, ha a 4., 6. és7. sorokat kikommentezzük? Indokoljuk a választ.

2. Feladat (4 pont) Válaszoljunk a következő kérdésekre az alábbi `./a.sh` UNIX Shell szkript végrehajtásával kapcsolatban.

```
 1  #!/bin/bash
 2
 3  RE='^(.*)([+-]) (-?[0-9]+) (-?[0-9]+)(.*)$'
 4  L="$1"
 5  while true; do
 6    if echo $L | grep -q -v -E "$RE"; then
 7      exit 1
 8    fi
 9
10    A=`echo $L | sed -E "s/$RE/\1/"`
11    B=`echo $L | sed -E "s/$RE/\3 \2 \4/"`
12    C=`echo $L | sed -E "s/$RE/\5/"`
13    L="$A `expr $B` $C"
14
15    echo "A='$A'  B='$B'  C='$C'"
16
17    if echo $L | grep -E -q "^ *-?[0-9]+ *$"; then
18      break
19    fi
20  done
21  echo $L
```

a) Mit fog a képernyőre írni az alábbi végrehajtás?

```
./a.sh "+ 1 2"
```

b) Mi lesz az `A`, `B` és `C` változók értéke a `while` ciklus minden egyes iterációjában és mit fog a képernyőre kiírni az alábbi végrehajtás?

```
./a.sh "+ + 1 2 3"
```

c) Mi lesz az `A`, `B` és `C` változók értéke a `while` ciklus minden egyes iterációjában és mit fog a képernyőre kiírni az alábbi parancssor? Indokoljuk a választ.

```
./a.sh "+ + -1 2 -3 4"; echo $?
```

d) Magyarázzuk meg részletesen a 17-19 sorok működését.

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p
Cerința **a) – 4.6p**
- Definirea clasei Sofa – **0.55p** din care
  - clasă abstractă – 0.05
  - atribute - 0.1
  - **constructor (a3)** - 0.1
  - metoda **setPrevious (a5)** – 0.1
  - metode **getCoefficient**, **getPrice**, **getNext**, **setNext**, **getPrevious** - 0.2
- Definirea clasei SofaWithRoudArms– **0.85p** din care
  - relația de moștenire – 0.2
  - atribut – 0.1
  - **constructor (a3)** – 0.2
  - metoda **getPrice (a2)** – 0.25
  - metoda **getCoefficient** - 0.1
- Definirea clasei ClassicSofa – **0.4p** din care
  - relația de moștenire – 0.25
  - metoda **getCoefficient (a5)** - 0.15
- Definirea clasei SellingProducts– **2.8p** din care
  - atribute– 0.2
  - **constructor (a3)** - 0.1
  - metoda **addProductToSell (a4)** – 2.5
    - adăugare listă vidă – 0.3
    - adăugare început – 0.3
    - adăugare sfârșit – 0.3
    - inserare în listă – 1.6

Funcția **b) - 1.1p**
- signatura - 0.1
- afișare în $\theta(n)$ - 1p
  * afișare în complexitate timp mai mare decât cea cerută– 0.25

Funcția **c) - 1.9p**
- signatura - 0.1
- ștergere în $O(n)$ – 1.8
  - ștergere început – 0.3
  - ștergere sfârșit – 0.3
  - ștergere în listă – 1.2

  * ștergere în complexitate timp mai mare decât cea cerută– 0.5

Funcția principală **d)** – **0.4p**
- construire obiecte – 0.2p
- apel funcție b) – 0.1p
- apel funcție c) - 0.1 p

Cerința **e)** – **1p**
- favorabil (0.25p)
- mediu (0.5p)
- defavorabil (0.25p)

BAREM INFORMATICĂ
VARIANTA 2

Subiect Baze de date

Oficiu – **1p**

Problema 1. Punctaj - 4p

- relații cu atribute corecte, chei primare, chei candidat: **3p**
- legături modelate corect (chei externe): **1p**

Problema 2. Punctaj - 5p

- **a** - rezolvarea completă a interogării: **2.5p**

- **b1** - rezultat evaluare interogare:

NumeParinte	Adresa
P2	A2
P3	A3

  - coloane – **0.5p**
  - valori tuplu – **1p**

- **b2** - {CodCopil}→{CodParinte} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodParinte}→{CodGrupa} nu este satisfăcută – **0.25p**; **0.25p** explicație

Notă: La specializările Informatică engleză și Informatică maghiară se iau în considerare versiunile traduse în limbile corespunzătoare.

BAREM INFORMATICĂ
VARIANTA 2

Subiect: Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) a b c d, pentru că sort citeşte din fa ce scrie programul şi apoi programul citeşte din fb ce afişează sort

1p – b) nicicum, cele trei procese fiu sunt concurente şi ordinea creării lor nu are nicciun efect asupra execuţiei

1p – c) x y z, pentru ca deşi redirecţionăm ieşirea standard în /dev/null, programul scrie în ieşirea de eroare

1p – d) nicicum, procesul fiu se încheie şi FIFO-ul e închis automat, astfel sort nu ramâne blocat la citire

1p – e) se va bloca fără a afişa nimic, pentru că nu poate deschide FIFO-urile, procesele fiu nemaifiind create din cauză că execlp suprascrie codul procesului părinte

Problema 2 (4p)

1p – a) 3 – scriptul calculează suma celor două numere

1p – b) A='+ ' B='1 + 2' C=' 3'
A=" B='3 + 3' C="
Se afișează 6

1p – c) A='+ ' B='-1 + 2' C=' -3 4'
A=" B='1 + -3' C=' 4'
Se va afişa 1 pentru că $L nu se va potrivi cu expresia regulară de la linia 6 şi se va face exit 1, iar $? va conţine această valoare

1p – d) Dacă variabila L conţine rezultatul final, adică o valoare întreagă cu eventuale spaţii înainte şi după, valoarea de adevăr a comenzii grep va fi TRUE şi ca urmare se intră in IF şi se va executa comanda break, astfel încheindu-se cu succes execuţia scriptului