BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2023. szeptember 5.
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

Megjegyzések

- A megfelelő programozási stílus hiánya (azonosítók beszédes elnevezése, tördelés, megjegyzések ha szükségesek, a kód olvashatósága) az adott tételhez tartozó pontszám 10%-nak az elvesztéséhez vezet.
- Pszeudokódban kizárólag a következő utasítások használata engedélyezett: *értékadás, minden, ha, amíg, ismételd, kiír* és *visszatérít*.
- Tilos új mezők és metódusok hozzáadása a kijelentésben említetteken kívül, leszámítva a konstruktorokat és destruktorokat. Tilos a kijelentésben megadott mezők és metódusok láthatóságának a megváltoztatása.

1. Feladat (1.5 pont)

Adott az $a[1], a[2]....a[n]$ ($n \geq 1$) különböző természetes számokat tartalmazó csökkenő sorozat és az x természetes szám. Írjatok algoritmust pszeudokódban, melynek időbonyolultsága $O(\log_2 n)$ és amely visszatéríti azt a *poz* pozíciót, amelyen az x elem megjelenik az a sorozatban (ha x megjelenik a sorozatban), vagy azt a pozíciót, amelyre x-et be kellene szúrni az a sorozatba úgy, hogy az elemek csökkenő sorrendje megmaradjon. Indokoljátok az algoritmus bonyolultságát. **Megjegyzés:** Egy elem beszúrása a k. pozicióra feltételezi a $k, k+1, ..., n$ pozíciókon lévő elemek egy pozícióval való jobbra tolását és a sorozat hosszának növelését.

Példa. Ha $n=4$ és az a sorozat (14, 12, 9, 5) akkor:

- ha $x=15$ a visszatérített érték $poz=1$, mivel a 15 beszúrása után a sorozat (**15**, 14, 12, 9, 5);
- ha $x=11$ a visszatérített érték $poz=3$, mivel a 11 beszúrása után a sorozat (14, 12, **11**, 9, 5);
- ha $x=12$ a visszatérített érték $poz=2$, mivel **12** már megjelenik a 2. pozíción.

2. Feladat (1.5 pont)

Írjatok algoritmust pszeudokódban, melynek bemenő paraméterei egy $x[1], x[2]....x[n]$ ($n \geq 1$) karakterlánc és egy k, $0 < k \leq n$ egész szám és amely felépíti azt az $y[1], y[2]....y[n]$ karakterláncot, mely az x karakterlánc körkörös permutációjának felel meg k pozícióval jobbra. Adjátok meg az algoritmus időbonyolultságát.

Példa. Ha $n=6$ és a karakterlánc elemei S, T, R, I, N, G, akkor $k=2$ esetén az y karakterlánc tartalma N,G,S,T,R,I.

3. Feladat (1.5 pont)

Adott egy n elemű a sorozat ($a[1], ..., a[n]$), mely az $\{1, ..., n\}$ ($n>0$) elemek egy permutációját jelöli. Írjatok algoritmust pszeudokódban, melynek időbonyolultsága a legrosszabb esetben $\theta(n)$ és amely meghatározza, hogy az a sorozat jelölheti-e egy olyan bináris keresőfa inorder bejárását, melynek n csúcsát 1-től n-ig számozzuk. Ha igen, akkor írjuk ki a 0 értéket, ellenkező esetben írjunk ki a felcseréléseknek egy listáját, melyek eredményeként egy olyan sorozatot kaphatunk, amely betartja a kért feltételt. Bármely megoldás amelynek időbonyolultsága nem tartozik a megadott bonyolultsági osztályhoz részleges pontszámot fog kapni.

Példa. Ha $n = 3$ és az a sorozat (3, 2, 1) akkor az 1 3 felcserélést írjuk ki (vagyis az 1 és 3 pozíciókon található elemeket cseréljük fel - így egy lehetséges fa, melynek inorder bejárása (1, 2, 3) az, melynek gyökere 2, bal gyereke 1 és jobb gyereke 3).

4. Feladat (2.25 pont)

Adott a következő (hiányos) C++ program. Egészítsétek ki a **Computer** és **SorterByTotalPrice** oszályok implementációját úgy, hogy a main függvény a „HC85, 165" üzenetet irja ki a képernyőre. A **SorterByTotalPrice** osztály **sort** metódusa termékek listáját kapja bemenetként, és a teljes ár szerint csökkenő sorrendbe rendezi ezeket.

A rendezést a standard C++ könyvtában található **std::sort** függvény meghívásával kell végrehajtani. Egy számítógép teljes ára az alapár és az adó összegeként számítódik ki.

```
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>

class Product
{
   private:
       std::string name;
       int baseprice;

   public:
       Product(const std::string& name, int baseprice):
                    name{name},
                    baseprice{baseprice} {}

       virtual int total_price() {
              return this->baseprice;
       }
       virtual std::string toString() {
              return this->name;
       }
      virtual ~Product() {}
};
class Computer: public Product {
   private:
       int tax;
       //A: missing code
};
class SorterByTotalPrice {
       //B: missing code
};
int main()
{
       std::vector<Product*> computers{new Computer{"HC90", 140, 10},
                                       new Computer{"HC91", 100, 12},
                                       new Computer{"HC85", 150, 15}};

       SorterByTotalPrice::sort(computers);
       std::cout << computers[0]->toString();
       for (auto c: computers)
           delete c;
       return 0;
}
```

5. Feladat (2.25 pont)

Implementáljátok C++ nyelvben az 1-3. pontokban kért osztályokat, és a 4. pontban kért függvényt:

1. A **Contact** absztrakt osztály, aminek van egy *name* karakterlánc típusú privát adattagja, egy **sendMessage** absztrakt (tiszta virtuális) metódusa egy karakterlánc típusú *message* paraméterrel, és egy **getName** metódusa, ami a *name* adattag értékét téríti vissza.
2. A **Person** osztály, ami a **Contact** osztályból származik, és van még egy *number* karakterlánc típusú privát adattagja. A **sendMessage** metódus paraméterként kap egy karakterlánc típusú üzenetet, és kiírja a személy nevét, számát, és a paraméterként kapott üzenetet.
3. A **Group** osztály, ami a **Contact** osztályból származik, és nulla vagy több kontaktot tartalmaz. Az **addContact** metódus paraméterként egy kontaktot kap, amelyet hozzáad a kontaktok listájához. A **sendMessage** metódus kiírja az összes kontakt nevét és számát, valamint a paraméterként kapott üzenetet.
4. Hozzátok létre a következő objektumokat (adjatok meg tetszőleges telefonszámokat *number* értékként): 2 személy „Mother" és „Father" nevekkel; 2 személy „Jane", „John" nevekkel; „Parents" csoport, ami a „Mother" és „Father" személyeket tartalmazza; „Family" csoport, ami tartalmazza a „Parents" csoportot és „Jane" személyt. A **sendMessage** függvényt felhasználva küldjétek el a következő üzenetet a „Family" csoportnak és „John" személynek: „You are invited to my birthday party next week!". A memóriakezelés helyes kell legyen.

Adatbázisok TÉTEL

1. Feladat (4 pont)

Egy kerékpárkölcsönző üzlet az alábbi információkat tárolja el egy relációs adatbázisban:

- Egy kerékpárgyártó cégről tároljuk: kód, megnevezés, származási ország.
- Egy kerékpár esetén tároljuk: kód, típus, méret (a méret egy szám, pl. 56), napi bérlés ára. Egy kerékpárt egy adott kerékpárgyártó cég gyárt.
- Egy kölcsönző esetén tároljuk: személyi szám (CNP), név, telefonszám, e-mail cím.
- Kerékpár(ok) kölcsönzésekor egy kölcsönzőlapot állítanak ki. Egy kölcsönző több kerékpárt is kibérelhet egyszerre. A kerékpárkölcsönző lap esetén nyilvántartjuk: kölcsönzőt, a kölcsönzés dátumát, valamint a kikölcsönzött kerékpárok listáját – minden kikölcsönzött kerékpár esetén külön tároljuk a kölcsönzés kezdetének-, illetve a visszahozatalának dátumát. Minden kölcsönzőnek egy adott napon egyetlen kölcsönzőlapot állítanak ki. Abban az esetben, ha egy kölcsönző a kölcsönzés napján visszatér az üzletbe és újabb kerékpárt kölcsönöz, ezen kerékpárokat a meglevő kölcsönzőlaphoz rendelik hozzá. Ugyanakkor ha egy kölcsönző egy másik napon látogat el az üzletbe, egy új kölcsönzőlapot állítanak ki számára.

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:

Utasok (UtasKod, UtasNev, Telefonszam)
Soforok (SoforKod, SoforNev, Fizetes)
Autok (AutoKod, Rendszam, Automarka, *SoforKod*)
Utazasok (UtazasKod, *UtasKod*, *AutoKod*, UtazasDatuma, IndulasIdopontja, UtazasHossza, IndulasiCim, Celallomas, Vegosszeg)

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, amely visszaadja azon sofőr(ök) kódját, nevét és fizetését, aki(k) fizetése a legnagyobb azon sofőrök között, akik legalább egy olyan utazást lebonyolítottak 'Ford' márkájú autóval, ahol a célállomás a 'Iulius Mall Cluj' volt! A lekérdezés szűrje ki a duplikátumokat!

b. Tekintsük az Autok és Utazasok relációk alábbi előfordulásait:

Autok:

AutoKod	Rendszam	Automarka	SoforKod
1	CJ11AAA	Ford	1
2	CJ22BBB	Opel	2
3	CJ33CCC	Volkswagen	3

Utazasok:

Utazas Kod	Utas Kod	Auto Kod	Utazas Datuma	Indulas Idopontja	Utazas Hossza	Indulasi Cim	Celallomas	Vegosszeg
1	1	1	2023.06.25	07:30	15	LP1	D1	20
2	1	2	2023.06.26	08:30	20	LP2	D3	30
3	2	1	2023.06.26	12:30	10	LP4	D4	15
4	1	1	2023.06.30	10:15	5	LP1	D2	10
5	3	3	2023.07.01	15:05	40	LP1	D4	50
6	3	2	2023.07.01	20:00	12	LP2	D2	15

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 a.SoforKod, COUNT(DISTINCT u.UtasKod) Szam
FROM Autok a INNER JOIN Utazasok u ON a.AutoKod = u.AutoKod
WHERE u.UtazasDatuma BETWEEN '2023.01.01' AND '2023.06.30'
GROUP BY a.SoforKod
HAVING COUNT(DISTINCT u.UtasKod) >= ALL
        (SELECT COUNT(DISTINCT u.UtasKod)
         FROM Autok a INNER JOIN Utazasok u ON a.AutoKod = u.AutoKod
         WHERE u.UtazasDatuma BETWEEN '2023.01.01' AND '2023.06.30'
         GROUP BY a.SoforKod)
```

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

- {UtazasKod} → {AutoKod}
- {AutoKod} → {IndulasiCim}.

Operációs rendszerek TÉTEL

1. Feladat (4 pont)
Válaszoljunk a következő kérdésekre, melyek az alábbi kódrészlet végrehajtására vonatkoznak, tudva, hogy az `mkfifo` függvény létrehoz egy FIFO állományt, az `unlink` függvény pedig törli azt!!

```
1  int main(int argc, char** argv) {
2      int f; char c = 'a';
3
4      mkfifo("fifo", 0600);
5      if(fork() == 0) {
6          f = open("fifo", O_RDONLY);
7          read(f, &c, 1);
8          printf("%c\n", c);
9          close(f);
10         exit(0);
11     }
12
13     f = open("fifo", O_WRONLY);
14     write(f, "xy", 2);
15     close(f);
16
17     wait(NULL);
18     unlink("fifo");
19     return 0;
20 }
```

a) Mit fog kiírni a képernyőre?

b) Mit fog kiírni a képernyőre, ha a 18. sort áttesszük a 9-10. sorok közé?

c) Hogyan befolyásolja a program működését, ha a 4. sort áttesszük az 5-6. sorok közé?

d) Milyen esetben fog 'a'-t írni?

e) Magyarázzuk el a 4. sorban levő `mkfifo` függvény paramétereinek a jelentését!

2. Feladat (5 pont)
Válaszoljunk a következő kérdésekre, melyek az alábbi UNIX Shell szkript végrehajtására vonatkoznak!

```
1  #!/bin/bash
2
3  for A in $*; do
4      for F in x/*; do
5          if ! grep -E -q "^$A(_+$A)*$" $F; then
6              echo "$A" >> $F
7          elif [ `grep -E "^$A(_+$A)*$" $F | wc -l` -gt 1 ]; then
8              grep -E -v "^$A(_{1,}$A){0,}$" $F > $F.aux
9              echo "$A" >> $F.aux
10             mv $F.aux $F
11         fi
12     done
13 done
```

a)	Mi lesz az `x/a.txt` állomány tartalma a szkript `abc` praméterrel történő futtatása után? (Az állomány eredeti tartalma a jobb oldali oszlopban szerepel.)	`abc` `def`

b)	Mi lesz az `x/b.txt` állomány tartalma a szkript `abc` praméterrel történő futtatása után? (Az állomány eredeti tartalma a jobb oldali oszlopban szerepel.)	`abc`
		`def`
		`abc`

c)	Mi lesz az `x/c.txt` állomány tartalma a szkript `abc def` praméterekkel történő futtatása után? (Az állomány eredeti tartalma a jobb oldali oszlopban szerepel.)	`def_def`
		`abc`
		`abc_def`
		`abc_abc`
		`def_def_def`

d)	Magyarázzuk meg részletesen a 7. sorban szereplő feltételt!	
e)	Magyarázzuk meg részletesen a 8. sorban szereplő reguláris kifejezést!	

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p

Problema 1. Punctaj - 1.5p

- Subalgoritm corect: **1p**
  - signatura corectă: 0.1p
  - implementare căutare binară: 0.9p
- Justificarea complexității (căutare binară): **0.5p**

Problema 2. Punctaj - 1.5p

- Subalgoritm corect implementat: **1.25p**
  - signatura corectă: 0.1p
  - implementare: 1.15p
- Complexitate $\theta(n)$: **0.25p**

Problema 3. Punctaj - 1.5p

Soluție: *șirul trebuie transformat în permutarea identică*.

- Soluție având complexitatea timp în caz defavorabil $\theta(n)$ - 1.5p
- Soluție având complexitatea timp în caz defavorabil $\theta(n \cdot \log_2 n)$ - 1p
- Soluție având complexitatea timp în caz defavorabil $\theta(n^2)$ - 0.75p
- Alte soluții care nu se încadrează în clasele de complexități de mai sus - 0.25p

Problema 4. Punctaj - 2.25p

- Implementarea clasei **SorterByTotalPrice** cu metoda statică **sort()** și folosirea funcției **std::sort**, cu o funcție lambda (**0.75p)**
  - pentru folosire **std::sort**, cu operator <
    bool operator<(Product*, Product*) 0.6p
  - pentru folosire **std::sort**, cu o functie standalone
    bool nume_functie(Product*, Product*) 0.75p
  - pentru funcții de sortare implementate (fără **std::sort**) corect 0.25p
- Implementarea metodei total_price in clasa **Computer** (**0.5p**)
- Implementarea constructorului în clasa **Computer** (**0.5p**)
- Implementarea metodei **toString()** in clasa **Computer** (**0.5p**)

Problema 5. Punctaj - 2.25p

- clasa ***Contact*** **(0.4p)**:
  - câmp privat *name*, constructor (0.1), metoda **getName()** (0.05), metoda abstractă **sendMessage** (0.15), destructor virtual (0.1).
- clasa **Person (0.35p):**
  - moștenire (0.05), câmp privat **number**, constructor (0.15), metoda **sendMessage** (0.15)
- clasa **Group (0.85p)**:
  - moștenire (0.05), câmp privat ***lista de contacte*** (0.2), constructor (0.1), metoda **addContact** (0.2), metoda **sendMessage** (0.3), destructor (a se vedea punctajul de mai jos pentru distrugerea corecta a obiectelor)
- main **(0.65p)**:
  - construire corectă obiecte **Person** (0.15), construire corectă obiecte **Group** + adăugare de contacte (0.3), apel **sendMessage** la Group si la Person (0.1), distrugere corecta obiecte (0.1)

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:

CodSofer	Nr
1	2

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

- **b2** - {CodCursa}→{CodAuto} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodAuto}→{LocPlecare} 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.

Subiect Sisteme de operare

Oficiu – 1p

Problema 1. Punctaj – 4p

a) Se va tipări 'x' – 0.5p
b) Se va tipări 'x' – 1p
c) Dacă părintele execută open înainte ca fiul să creeze fifo, fiul se blochează la open şi părintele la wait – 1p
d) Niciodată – 1p
e) Primul este numele fifo-ului de creat şi al doilea sunt permisiunile de acces – 0.5p

Problema 2. Punctaj – 5p

a) Conţinutul va fi acelaşi – 1p
b) Conţinutul va fi `def` urmat de `abc` pe altă linie – 1p
c) Conţinutul va fi `abc_def` urmat de `abc` pe altă linie urmat de `def` pe altă linie – 1p
d) Fişierul `$F` conţine cel puţin 2 linii formate exclusiv din una sau mai multe valori `$A` separate prin cel puţin câte un _ - 1p
e) Întreaga linie e o secvenţă care începe cu valoarea `$A` urmată de 0 sau mai multe secvenţe formate din unul sau mai multe _ urmate de valoarea `$A` – 1p