BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2023 július 3
Informatika - magyar nyelv

1. 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 kívűl, leszámítva a konstruktorokat és destruktorokat. Tilos a kijelentésben megadott mezők és metódusok láthatóságának a megváltoztatása.

1. Feladat (1.5 pont)

Tekintsük az alábbi pszeudokódban megadott algoritmusokat. A **transformare** algoritmus bemenő paraméterei az n elemű egész számokat tartalmazó x sorozat ($x[1], \ldots, x[n]$) és az m elemű egész számokat tartalmazó y sorozat ($y[1], \ldots, y[m]$) és kimenő paraméterei a k elemű egész számokat tartalmazó z sorozat ($z[1], \ldots, z[k]$). A **g** algoritmus bemenő paraméterei az n elemű egész számokat tartalmazó x sorozat ($x[1], \ldots, x[n]$) és az i természetes szám ($1 \leq i \leq n$) és kimenő paraméterei az n elemű egész számokat tartalmazó y sorozat ($y[1], \ldots, y[n]$). Mit fog tartalmazni a z sorozat a **transformare**($x, n, y, 0, z, k$) hívás eredményeképpen? Indokoljátok a választ.

```
Algoritmus g(x, n, y, i)
  Ha i ≤ n akkor
    y[i] ← x[i]
    g(x, n, y, i + 1)
  Vége(Ha)
Vége(Algoritmus)

Algoritmus transformare(x, n, y, m, z, k)
  Ha n = 0 akkor
      g(y, m, z, 1)
      k ← m
    egyébként
      y[m +1] ← x[n]
      transformare(x, n - 1, y, m + 1, z, k)
  Vége(Ha)
Vége(Algoritmus)
```

2. Feladat (3 pont)

Adott az $a[1], \ldots, a[n]$ sorozat, mely n ($4 \leq n \leq 10000$) egész számot tartalmaz a $[-10^{15}, 10^{15}]$ intervallumból. Írjatok olyan programot C++ nyelvben, melynek időbonyolultsága $O(n^3)$ a legrosszabb esetben és amely a képernyőre négy x, y, z, t ($x, y, z, t \in [1, n]$, $x < y < z < t$) természetes számot ír ki, úgy, hogy $a[x] + a[y] + a[z] + a[t] = 0$. Abban az esetben ha nem létezik a sorozatban négy különböző pozíción található elem melyek összege 0, a képernyőre a -1 értéket írjuk ki. **Megjegyzés:** *Bármely megoldás amelynek időbonyolultsága nem tartozik a megadott bonyolultsági osztályhoz részleges pontszámot fog kapni. Az STL-ben található algoritmusok és adatszerkezetek használata engedélyezett.*

3. Feladat (2.25 pont)

Implementáljátok C++ nyelvben a **MyObject, MyInteger, MyString, MyObjectlist** és **MyListIterator** osztályokat úgy, hogy a következő függvény végrehajtása után a kommentben megadott kimenet jelenjen meg, és a memória helyesen legyen kezelve.

```
void function()
{
	MyObjectList list{};
	list.add(new MyInteger{ 2 }).add(new MyString{ "Hi" });
	MyString* s = new MyString{ "Bye" };
	list.add(s).add(new MyString{ "5" });

	MyListIterator i{ list };
	while (i.isValid()) {
		MyObject* o = i.element();
		o->print();
		i.next();
	} // prints: 2 Hi Bye 5
}
```

4. Feladat (2.25 pont)

Implementáljátok C++ nyelvben a **Szemely** osztályt, ami a következő privát adattagokkal rendelkezik: *csaladnev* (karakterlánc), *keresztnev* (karakterlánc), *eletkor* (egész szám). Az osztálynak implementáljatok egy konstruktort, amely inicializálja az összes adattagot, illetve az adattagokhoz való hozzáférést biztosító metódusokat. Implementáljatok egy függvényt, ami személyek listáját rendezi. A függvény bemeneti paraméterként megkapja a **Szemely** típusú objektumok listáját, valamint egy rendezési kritériumot reprezentáló függvényt. A rendezési algoritmus időbonyolultsága a legrosszabb esetben $\theta(n \log_2 n)$ kell legyen, ahol n a lista hossza.

Hívjátok meg a függvényt, hogy növekvő sorrendbe rendezzétek a **Szemely** típusú objektumok listáját:

- *csaladnev* adattag szerint
- *eletkor* adattag szerint
- *eletkor* és *csaladnev* adattagok szerint (amennyiben a két személy *életkora* megegyezik, a *családnév* szerint lesznek rendezve)

Megjegyzés. Nem használhatóak beépített rendezési algoritmusok.

Adatbázisok TÉTEL

1. Feladat (4 pont)

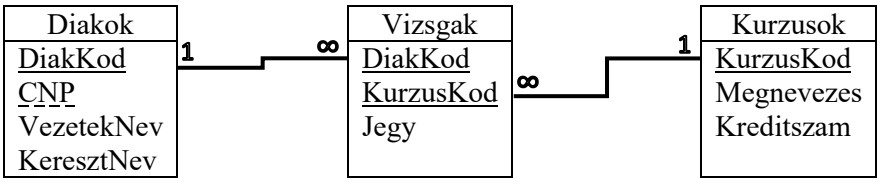
Egy műkorcsolya bajnokságon részt vevő versenyzők különböző versenyszámokban indulhatnak. A versenyszámok során a versenyzőket versenybírák értékelik. Az adatokat egy relációs adatbázisban tároljuk.

- Egy versenyző esetén tároljuk: versenyző kódját, vezetéknevét, keresztnevét, születési dátumát.
- Egy versenyszám esetén tároljuk: versenyszám kódját, megnevezését, leírását, dátumát.
- Egy versenyző több versenyszámban is indulhat, ugyanakkor egy versenyszámra is több versenyző iratkozhat fel.
- Egy versenybíróról tároljuk: versenybíró kódját, vezetéknevét, keresztnevét.
- Egy versenybíró egy adott versenyzőnek egy adott versenyszámban elért teljesítményét egy pontszámmal értékeli (a pontszám 1 és 10 közötti érték lehet). Egy versenybíró több versenyzőt is értékelhet, akik egyazon versenyszámban indulnak, illetve ugyanazon versenyzőt értékelhet több különböző versenyszámban is. Egy versenybíró egy adott versenyző egy adott versenyszámban történő indulását legtöbb egy alkalommal értékelheti. Egy versenyző csak azon versenyszámokban érhet el pontot, amelyekben elindult/amelyekre feliratkozott.

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:

Ujsagok[UjsagKod, UjsagNev]
Ujsagirok[UjsagiroKod, Vezeteknev, Keresztnev, *UjsagKod*]
Kategoriak[KategoriaKod, Megnevezes]
Cikkek[CikkKod, Cim, MegjelenesiDatum, Szoveg, *UjsagiroKod*, *KategoriaKod*]
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 megadja azon cikkek számát, amelyek a *Gazdaság* megnevezésű kategóriába tartoznak és az *Universul* nevű újság újságírói által íródtak.

b. Tekintsük a Kategoriak és Cikkek relációk alábbi előfordulásait:

Kategoriak

KategoriaKod	Megnevezes
1	c1
2	c2
3	c3

Cikkek

CikkKod	Cim	MegjelenesiDatum	Szoveg	UjsagiroKod	KategoriaKod
1	a1	2022-1-1	t1	1	2
2	a2	2022-2-1	t2	2	3
3	a3	2022-1-1	t3	3	1
4	a4	2022-5-1	t4	1	2
5	a5	2022-3-1	t5	2	1
7	a7	2022-7-1	t7	2	2
8	a8	2022-5-1	t8	4	2

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 c.UjsagiroKod, COUNT(*) Szam
FROM Cikkek c INNER JOIN Kategoriak k ON c.KategoriaKod = k.KategoriaKod
WHERE k.Megnevezes = 'c2'
GROUP BY c.UjsagiroKod
HAVING 2 < (SELECT COUNT(*)
            FROM Cikkek c2
            WHERE c2.UjsagiroKod = c.UjsagiroKod)
```

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

- $\{UjsagiroKod\} \rightarrow \{Cim\}$
- $\{CikkKod\} \rightarrow \{Szoveg\}$

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 1

Subiect Algoritmică şi Programare

Oficiu – 1p

Problema 1. Punctaj - 1.5p

- Răspuns corect (șirul *z* va conține elementele șirului *x* în ordine inversă): **0.75 p**
- Justificare: **0.75p**

Problema 2. Punctaj - 3p

- Soluție având complexitatea timp în caz defavorabil $O(n^3)$ - 3p
- Soluție având complexitatea timp în caz mediu $\theta(n^3)$ - 1.75p
- Soluție având complexitatea timp $O(n^3 \log_2 n)$ - 1.25p
- Soluție având complexitatea timp $O(n^4)$ - 0.5p

Problema 3. Punctaj - 2.25p

- Clasa **MyObject** (**0.2p**): abstractă, funcția abstractă **print,** virtual destructor
- Clasa **MyInteger** (**0.3p):** mostenire, camp privat de tip *int* + constructor + funcția **print**
- Clasa **MyString (0.3p)**: mostenire, camp privat de tip *string* + constructor, funcția **print**
- Clasa **MyObjectList (0.85p)**:
  - vector de elemente *Object* (pointers) (0.2), funcția **add** (0.2), funcția **length** (0.1), destructor (0.15), funcție care accesează elementul de pe o poziția dată *sau* funcție **getObjects** *sau* friend class MyListIterator (0.2).
- Clasa My**ListIterator (0.6p)**:
  - câmpuri private de tip *MyObjectList* și *int* (elementul curent) + constructor (0.25), funcția **isValid** (0.1), funcția **next** (0.1), funcția **element** (0.15).

Problema 4. Punctaj - 2.25p

- Clasa Persoana - **0.3p**
- Signatura funcției: lista/array + parametru de tip funcție (de comparare sau funcție care returnează cheia după care sortăm) - **0.3p**
- Implementare **Merge Sort** (partea recursivă/interclasare) - 0.6+0.75 = **1.35p**
- Implementare funcții de comparare/cheie (sau funcții lambda) și apel corect pentru cele 3 variante de sortare- **0.3p**

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:

CodJurnalist	Nr
2	1

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

- **b2** - {CodJurnalist}→{Titlu} nu este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodArticol}→{Text} 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 – 5p

a) Previne procesul fiu de la a citi din `a` ceea ce a scris el însuşi – 1p
b) Se va afişa 0 şi 3 în orice ordine – 1p
c) Dacă procesul fiu citeşte din `a` ce a scris chiar el, se blochează, altfel functionează ca la a) – 1p
d) Funcţionarea nu este afectată – 1p
e) Procesul fiu îşi continua execuţia cu instricţiunile de după IF – 1p

Problema 2. Punctaj – 4p

a) Numele fişierelor normale din directorul /usr/bin găsite recursiv şi în subdirectoare – 0.5p
b) La început de linie, minus urmat de orice grup de caractere de lungime 2, 5 sau 8 urmat de x – 1p
c) Fişierele care nu pot fi executate de utilizatorul care rulează scriptul – 0.5p
d) Fişierele care au drept de execuţie pentru owner, group sau others – 1p
e) Elimină din lista de procese cele care conţin în linia de comandă `$F` pe cele care conţin cuvântul `grep` – 1p