BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA KAR

Licenszvizsga 2026
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.

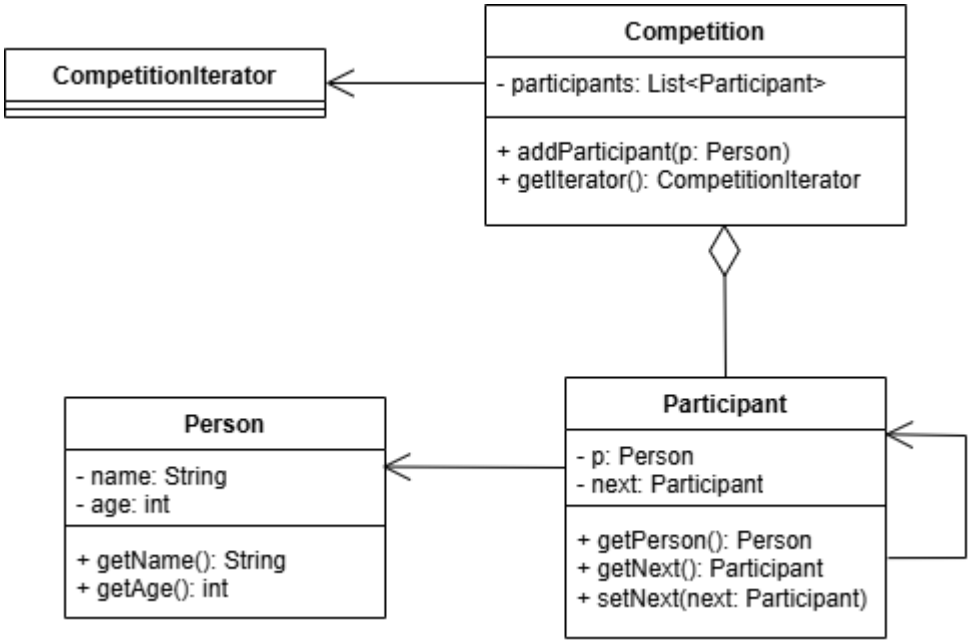
Algoritmusok és programozás TÉTEL

- **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 (Python, C++, Java, C#) létező könyvtárai.**

Egy versenyre 7 korosztályból iratkoznak be résztvevők: c_0 (5-9 év), c_1 (10-19 év), c_2 (20-29 év), c_3 (30-39 év), c_4 (40-49 év), c_5 (50-59 év) és c_6 (60-69 év). A résztvevők korosztályok szerint c_0, c_1, …, c_6 sorrendben és minden egyes korosztályon belül ábécésorrendben lesznek eltárolva.

1. **(2.5 pont)** Írjatok függvényt a **Python**, **C++**, **Java** vagy **C#** programozási nyelvek valamelyikében, amely paraméterként megkap egy *lst* listát, amely 7 darab, ábécésorrendbe rendezett karakterláncokat tartalmazó listából áll (melyek egy verseny résztvevőinek neveit ábrázolják korosztályok szerint, ahol a *lst* lista *i.* pozícióján a c_i korosztályhoz tartozó résztvevők nevei találhatóak), egy *varsta* természetes számot és egy *nume* karakterláncot (amelyek egy résztvevő életkorát és nevét tárolják). A függvény be kell szúrja a résztvevő nevét a korosztálynak megfelelő listába. A beszúrás olyan módon kell történjen, hogy ez a lista ábécésorrendben rendezett maradjon. ***Tilos előre definiált beszúró függvények használata egy elemnek a lista belsejébe történő beszúrására.***
2. **(1 pont)** Adjátok meg az 1-es pontbeli függvény időbonyolultságát a *legjobb*, *átlag* és *legrosszabb* esetben. Indokoljátok a választ.
3. **(5.5 pont)** Adott az alábbi UML diagram, mely tartalmazza a **Competition** (Verseny), **Participant** (Résztvevő), **Person** (Személy) és **CompetitionIterator** (VersenyIterator) osztályokat.

Notă ***Az osztályok konstruktorai nincsenek feltüntetve a diagramon.***



- A **Competition** osztály rendelkezik egy *participants* adattaggal, amely egy a korosztályok számával megegyező méretű lista. A *participants* lista *i*-edik pozícióján a c_i korosztályhoz tartozó résztvevők vannak eltárolva. Az osztály a következő metódusokkal rendelkezik:
  - **addParticipant(p: Person)**, amely hozzáadja a **p** személyt a korosztályának megfelelő listához.
  - **getIterator()**, amely visszatérít egy iterátort a verseny résztvevőinek bejárásához (iterálásához).
- A **Participant** osztály egy személyt tárol a versenyen egy adott korosztályban részt vevő személyek listájából. Az egyes korosztályok résztvevői a neveik ábécésorrendjében vannak eltárolva.
- A **Person** osztály egy versenyen részt vevő személy nevét (*name*) és életkorát (*age*) tárolja el. Az osztály rendelkezik metódusokkal az adattagok visszatérítésére. A személyek életkora egy a [5-69] intervallumba eső természetes szám.
- A **CompetitionIterator** osztály egy iterátort definiál a versenyen részt vevő személyek bejárására, azok korosztályainak sorrendjében (c_0, c_1, …, c_6), egy korosztályon belül pedig a nevek ábécésorrendjében.

Írjatok programot a **C++**, **Java** vagy **C#** programozási nyelvek valamelyikében, az alábbi követelmények szerint:

a) Deklaráljátok a **Competition**, **Person** és **Participant** osztályokat, azok adattagjait és metódusait a fenti diagramnak megfelelően. Csak a következő metódusokat kell implementálni:
  a1) A **Competition** osztály konstruktorát.
  a2) A **Participant** osztály **setNext**(**p**: **Participant**) metódusát.

b) Definiáljatok egy **CompetitionIterator** osztályt, beleértve az iterátorra jellemző adattagokat és metódusokat. Implementáljátok a **CompetitionIterator** osztály összes metódusát. Az implementált adattagok és metódusok $\theta(1)$ memóriabonyolultságúak kell legyenek. A metódusok időbonyolultsága $\theta(1)$ kell legyen.

c) Definiáljatok egy függvényt, amely paraméterként egy **Competition** típusú **c** objektumot kap, és a b) pontban definiált osztály segítségével kiírja a verseny résztvevőit a korosztályaik sorrendjében, egy korosztályon belül pedig a nevek ábécésorrendjében.

d) Definiáljatok egy függvényt, amely paraméterként egy **Competition** típusú **c** objektumot és egy *name* karakterláncot kap, és visszatéríti azt a korosztályt, amelyben a *name* nevű személy a versenyen részt vesz, vagy az üres karakterláncot abban az esetben, ha nem létezik egyetlen résztvevő sem a *name* névvel. Ha több *name* nevű személy létezik, térítsük vissza az első olyan korosztályt, amelyben ilyen nevű személy vesz részt.

e) Hozzatok létre egy **Competition** típusú **o** objektumot, amelyhez adjatok hozzá 2 korosztályt és 4 résztvevőt (mindegyik korosztályba 2-t). Ezt követően hívjátok meg a c) pontban definiált függvényt. Az egyik résztvevő személy esetén hívjátok meg a d) pontban definiált függvényt is.

Adatbázisok TÉTEL **2. VÁLTOZAT**

1. Feladat (4 pont)

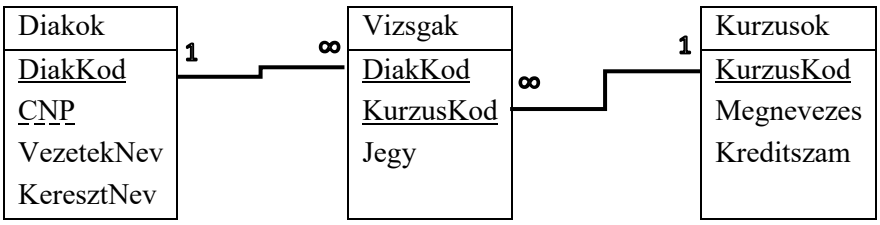
Egy médiacég relációs adatbázist használ tevékenységeinek kezelésére.

- Minden újság esetén érdekel: egyedi azonosító kódja, neve (amely minden újság esetén egyedi), alapítási éve, illetve megjelenési gyakorisága (ez lehet egyszerű attribútum vagy karakterlánc; pl. *napi, heti, havi*). Minden újságot egy adott központi iroda menedzsel, de egy adott központi iroda több újságot is menedzselhet.
- Minden központi iroda esetén tároljuk: egyedi azonosító kódját és telefonszámát (amely minden központi iroda esetén egyedi).
- Minden újságíró esetén érdekel: egyedi azonosító kódja, vezetékneve, keresztneve, születési dátuma, neme és pályakezdésének éve. Egy újságíró több újsággal is állhat együttműködési szerződésben, míg egy újság több újságíróval is együttműködhet. Egy adott újságíró egy adott újsággal csak egy együttműködési szerződést köthet. Minden együttműködési szerződés esetén tároljuk: az újságot, az újságírót, szerződéskötés dátumát, szerződés időtartamát (hónapok számát jelölő egész szám), valamint a javadalmazást.
- Minden újságcikknek van: egyedi azonosító kódja és címe. Az újságcikket egy újságíró írja valamely olyan újság számára, amellyel együttműködési szerződésben áll.
- Minden téma esetén tároljuk: egyedi azonosító kódját, megnevezését (amely egyedi minden téma esetén; pl. *Gazdaság, Sport*) és leírását. Egy adott újságcikk több témával is foglalkozhat, míg egy adott téma több újságcikkben is megjelenhet.

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:

Tartalomgyartok[TartalomgyartoKod, Nev, Orszag, SzuletesiEv]
Videok[VideoKod, Cim, Kategoria, Idotartam, *TartalomgyartoKod*]
Platformok[PlatformKod, Megnevezes]
VideokPlatformjai[*VideoKod, PlatformKod*, Monetizalas, KozzetetelEve]

Az elsődleges kulcsok alá vannak húzva, míg a külső kulcsok dőlt betűvel vannak szedve, nevük pedig megegyezik azon attribútumok nevével, amelyekre hivatkoznak.

a. Írjunk egy SQL lekérdezést, mely megadja minden olyan tartalomgyártó kódját és nevét, aki teljesíti az alábbi két feltételt:

1. ugyanabban az évben legalább 2 videót tett közzé, valamint
2. legalább 1 videót tett közzé a "StreamNow" nevű platformon.

Megjegyezzük, hogy egy videó *közzé van téve*, ha megjelenik egy platformon.

Példa az 1.feltételre:
- a T1 tartalomgyártó közzétett videói: V1 2020-ban, V2 2020-ban, V3 pedig 2021-ben lett közzétéve; T1 tehát teljesíti az 1. feltételt;
- a T2 tartalomgyártó közzétett videói: V4 2020-ban, V5 2021-ben, V6 pedig 2022-ben lett közzétéve; T2 tehát nem teljesíti az 1. feltételt.

b. Tekintsük a Tartalomgyartok, Videok, Platformok és VideokPlatformjai relációk alábbi előfordulásait:

Tartalomgyartok

Tartalom-gyartoKod	Nev	Orszag	SzuletesiEv
1	n1	Románia	2005
2	n2	Románia	2005
3	n3	Franciaország	2005

Platformok

Platform Kod	Megnevezes
1	p1
2	p2
3	p3

Videok

Video Kod	Cim	Kategoria	Idotartam	Tartalom-gyartoKod
1	v1	k1	3	1
2	v2	k2	3	1
3	v3	k1	4	1
4	v4	k1	2	2
5	v5	k1	3	2
6	v6	k2	4	1
7	v7	k2	2	1

VideokPlatformjai

Video Kod	Platform Kod	Monetizalas	Kozzetetel Eve
1	2	T	2020
2	2	F	2020
3	3	T	2021
4	3	T	2020
5	3	T	2021
6	1	F	2020
7	1	F	2021

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 vp.PlatformKod, MIN(vp.KozzetetelEve) MinEv
FROM Tartalomgyartok tgy INNER JOIN Videok v ON tgy.TartalomgyartoKod = v.TartalomgyartoKod
     INNER JOIN VideokPlatformjai vp ON v.VideoKod = vp.VideoKod
WHERE tgy.Orszag = 'Románia'
GROUP BY vp.PlatformKod
HAVING COUNT(vp.VideoKod) >
     (SELECT COUNT(*)
     FROM VideokPlatformjai vp2 INNER JOIN Platformok p2
          ON vp2.PlatformKod = p2.PlatformKod
     WHERE p2.Megnevezes = 'p2')
```

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

- $\{Kategoria\} \rightarrow \{Cim\}$
- $\{TartalomgyartoKod\} \rightarrow \{VideoKod\}$

Operációs rendszerek TÉTEL

1. feladat (5 pont) Válaszoljunk az `./a.out` program végrehajtásával kapcsolatos alábbi kérdésekre, tudva, hogy az az itt megadott forráskódból lett lefordítva. Feltételezzük, hogy minden szükséges fejlécállomány jelen van, továbbá a `fork` és `write` rendszerhívások sikeresen hajtódnak végre. Megjegyezzük, hogy a `WEXITSTATUS(st)` makró kinyeri a gyerekfolyamat által az `exit`-en keresztül közvetített kilépési kódot (a 0–255 intervallumban levő 8 bites egész).

```
 1  int main(int argc,
 2           char** argv){
 3    int i, st, total = 0;
 4    for (i=1; i<argc; i++) {
 5      if (fork() == 0) {
 6        exit(strlen(argv[i]));
 7      }
 8      wait(&st);
 9      total += WEXITSTATUS(st);
10    }
11    printf("%d\n", total);
12    return 0;
13  }
```

a) Mit fog kiírni az `./a.out ab cde f` ? Indokoljuk a választ.

b) Hány gyerekfolyamatot hoz létre az `./a.out ab cde f` ? Hát az `./a.out` (paraméterek nélkül), és ebben az esetben hány gyerekfolyamat jön létre? Indokoljuk a választ.

c) Magyarázzuk részletesen a `6.` sor működését, és azt, hogy hogyan jut el az `exit` által közvetített érték a szülő folyamathoz?

d) Részletezzük a `8-9.` sorok működését.

e) Mit ír ki az `./a.out`, ha egy 260 karakteres argumentummal hívjuk meg? Indokoljuk a választ.

2. feladat (4 pont) Válaszoljunk az alábbi `./a.sh` nevű UNIX Shell szkript végrehajtásával kapcsolatos kérdésekre, feltéve, hogy a megadott `d.txt` állományt használja. Megjegyzés: a sorszámok NEM tartoznak az állományhoz.

```
 1  #!/bin/bash
 2  while read L; do
 3    N=0;
 4    for X in `echo "$L" | sed -E "s/(.)/\1 /g"`; do
 5         N=`expr $N + 1`
 6    done
 7    if [ $N -ge 4 ]; then
 8         echo "$L" | sed -E "s/^(..).*$/\1/g"
 9    else
10         echo "$N"
11    fi
12  done < d.txt
```

`d.txt`

```
1  hello
2  ab
3  sun
4  code
```

a) Határozzuk meg, hogy mit ír ki a szkript az adott `d.txt` állomány 1. illetve 2. sorára. Indokoljuk a választ.
b) Magyarázzuk el a 8. sorban levő `sed` parancs működését.
c) Határozzuk meg, hogy milyen numerikus értékek lesznek kiírva. Indokoljuk a választ.
d) Részletezzük a `3-6.` sorok működését.

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1. – 2.5p**
- signatura – 0.2p
- implementare – **2.3p** din care
  - căutarea listei corespunzătoare vârstei *varsta* – 0.3p
  - adaugarea numelui *nume* în lista identificată – 2p

Cerința **2. – 1p**
- caz favorabil **0.2p** din care
  - complexitate – 0.1
  - justificare – 0.1
- caz mediu **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2
- caz defavorabil **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2

Cerința **3.a) – 1p**

Definirea clasei Competition – **0.4p** din care
- atribut – 0.1
- **constructor (a1)** – 0.2p
- metode **addParticipant**, **getIterator** – 0.1

Definirea clasei Participant – **0.3p** din care
- atribute – 0.1
- metode **getPerson, getNext** – 0.1
- **metoda setNext (a2)** – 0.1p

Definirea clasei Person– **0.3p** din care
- atribute – 0.1
- constructor – 0.1
- metode **getName, getAge** – 0.1

Cerința **3.b) – 2.5p**

Definirea clasei CompetitionIterator
- atribute – 0.5
- constructor – 0.5
- metode specifice - 1.5

Funcția **3.c)** – **0.7p**
- signatura – **0.1p**
- parcurgere – **0.6p**

Funcția **3.d)** – **1p**
- signatura – **0.1p**
- implementare funcție – **0.8p**
- returnare rezultat – **0.1p**

Funcția principală **3.e)** – **0.3p**
- construire obiect **o** – 0.1p
- adăugare 2 categorii de vârstă și 4 participanți – 0.1p
- apel funcții c) și d) – 0.1p

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:

CodPlatforma	AnMin
3	2020

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

- **b2** - {Categorie}→{Titlu} nu este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodCreator}→{CodVideo} 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 INFORMATICA
VARIANTA 2

Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) 6 = strlen("ab") + strlen("cde") + strlen("f") = 2 + 3 + 1

1p – b) 3 fii pentru ./a.out ab cde f; 0 fii pentru ./a.out fără argumente, și se afișează 0.

1p – c) Linia 5 termină procesul fiu cu un cod de ieșire egal cu lungimea argumentului argv[i]; valoarea ajunge la părinte prin starea de terminare citită de wait.

1p – d) `wait(&st)` aşteaptă terminarea fiului curent şi pune starea lui în st; `WEXITSTATUS(st)` extrage codul de ieşire (lungimea), care se adună în `total`.

1p – e) Afişează 4; nu va afișa 260, deoarece codul de ieşire ocupă doar 8 biţi (0–255);

Problema 2 (4p)

1p – a) (linia 1: hello) → he, (linia 2: ab) → 2. …+ justificare

1p – b) Comanda sed înlocuiește fiecare linie (aici vom avea una singură) cu primele două caractere ale ei, ignorând restul textului.

1p – c) se va afișa 2 și 3 (pentru liniile 2 și 3 din `d.txt`)

1p – d) Liniile 3-6 calculează în N **lungimea** liniei curente