UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Proba scrisă a examenului de licenţă 2026
Informatică Română

VARIANTA 1

NOTĂ

- Toate subiectele sunt obligatorii. La toate subiectele se cer rezolvări cu soluții complete.
- Nota minimă ce asigură promovarea este 5.00.
- Timpul efectiv de lucru este de 3 ore.

SUBIECT Algoritmică și programare

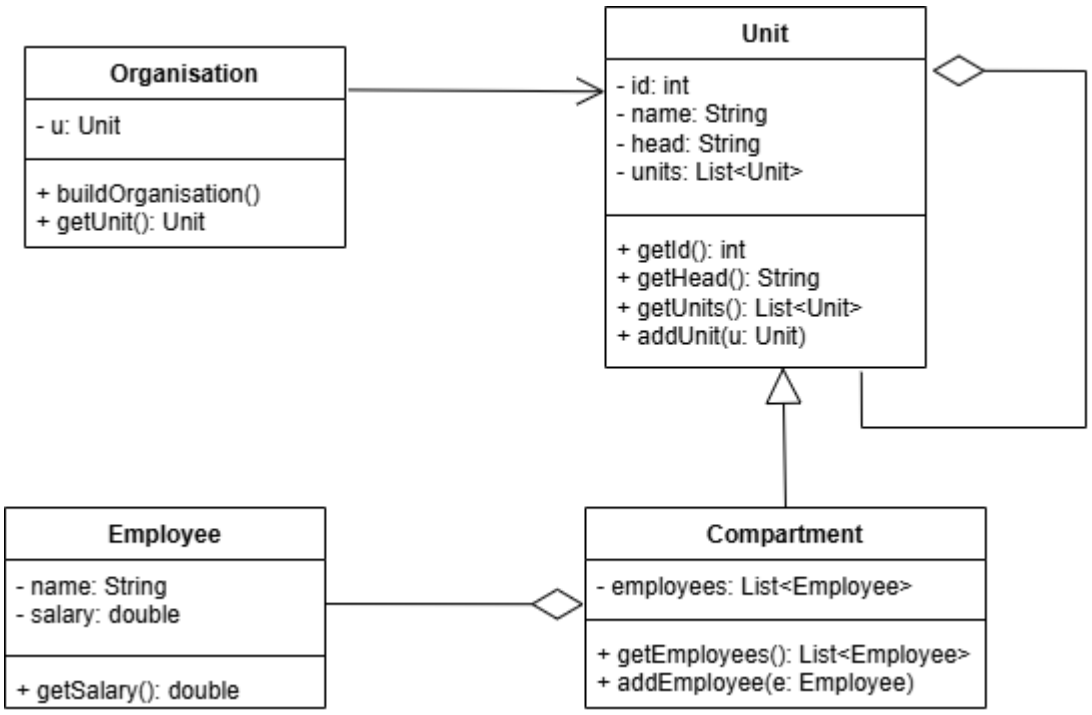
- **Se va indica limbajul de programare folosit.**
- **În implementările cerute nu este necesară dealocarea zonelor de memorie alocate dinamic.**
- **Lipsa unui stil de programare adecvat (denumiri sugestive pentru variabile, indentarea codului, comentarii, dacă sunt necesare, lizibilitatea codului) duce la pierderea a 10% din punctajul aferent subiectului.**
- **Nu adăugați alte atribute, metode, în afara celor menționate în enunț, cu excepția constructorilor și a destructorilor, dacă e cazul. Nu modificați vizibilitatea atributelor specificate în enunț.**
- **Nu se vor folosi containere sortate, operații de sortare sau căutare predefinite.**
- **Pentru tipurile de date puteți folosi biblioteci existente (Python, C++, Java, C#).**

1. **(3 puncte)** Scrieți o funcție într-unul dintre limbajele de programare **Python, C++, Java** sau **C#**, care primește ca parametri un șir ordonat descrescător *v* de numere reale (reprezentând salariile unor angajați din cadrul unui compartiment) și un număr real *salariu* (reprezentând salariul unui angajat nou). Funcția va insera *salariu* în șirul *v*, astfel încât acesta să rămână ordonat descrescător după salariul angajaților. Se va folosi o funcție auxiliară implementată iterativ, care returnează, într-o complexitate timp $O(\log_2 n)$, poziția pe care *salariu* ar trebui inserat în șirul *v*, *n* fiind lungimea șirului *v. Soluțiile recursive se punctează parțial*.
   Notă. Inserarea unui element pe o poziție *k* în șirul *v* presupune deplasarea elementelor de pe pozițiile *k, k+1*,..., *n* cu o poziție spre dreapta, în urma căreia lungimea șirului crește cu 1. ***Nu se vor folosi funcții de inserare predefinite în biblioteci pentru inserarea unui element pe o poziție în interiorul șirului.***

2. **(1 punct)** Precizați complexitatea timp în cazurile *favorabil, mediu* și *defavorabil* pentru funcția de la punctul **1**. Justificați răspunsul.

3. **(5 puncte)** Se dă următoarea diagramă UML conținând clasele **Organisation** (Organizație, reprezintă structura ierarhică a unei organizații), **Unit** (Unitate organizațională), **Compartment** (Compartiment), **Employee** (Angajat).

Notă: ***Constructorii claselor nu sunt indicați pe diagramă.***



- Clasa **Organisation** reprezintă o organizație (reprezentată sub forma unei structuri ierarhice de unități organizaționale). Metoda **getUnit**() returnează atributul *u* al clasei.
  - Metoda **buildOrganisation**() creează structura ierarhică a organizației. *Această metodă nu va trebui implementată.*
- Clasa **Unit** reprezintă structura ierarhică a unei unități organizaționale. Aceasta are un identificator (*id*), un nume (*name*) și un director (al cărui nume este memorat în atributul *head*). Unitatea are în subordine directă mai multe (sub)unități (memorate în atributul *units*). Clasa are metode pentru accesarea *id*, *head* și *units* și o metodă **addUnit**(*u*: **Unit**) care adaugă unitatea *u* la finalul listei *units*.
- Clasa **Compartment** reprezintă o unitate care nu mai are în subordine alte (sub)unități. Aceasta memorează în atributul *employees* o listă de angajați (**Employee**). Lista de angajați *employees* este memorată în ordine descrescătoare a salariilor acestora. Clasa are o metodă **addEmployee**(*e*: **Employee**) care adaugă un angajat în lista ordonată *employees* și o metodă **getEmployees**() care returnează lista angajaților din compartiment.
- Clasa **Employee** reprezintă un angajat caracterizat printr-un nume (*name*) și salariu (*salary*). Clasa are o metodă **getSalary**() care returnează atributul *salary*.

Notă: *Unitățile* (**Unit**) *din cadrul organizației* (**Organisation**) *au identificatori (id) distincți.*

Scrieți un program într-unul din limbajele de programare **C++, Java**, sau **C#**, cu următoarele cerințe:

a) Declarați toate clasele, atributele și metodele conform diagramei de mai sus. Implementați doar constructorul clasei **Compartment**.

b) Definiți o funcție care primește ca parametru un obiect *o* de tip **Organisation** și un număr întreg *n* și returnează o listă conținând toți angajații subordonați (direct sau indirect) unității din organizație având *id*-ul egal cu *n*. Lista returnată va fi vidă în cazul în care organizația nu are o unitate având *id*-ul egal cu *n*.

c) Definiți o funcție care primește ca parametru un obiect *o* de tip **Organisation** și care returnează lista directorilor de compartimente din organizația *o*.

d) Construiți un obiect *o* de tip **Organisation**, apelați metoda **buildOrganisation()**,apoi apelați funcțiile de la **b)** și **c)** pentru organizația creată.

SUBIECT Baze de date **VARIANTA 1**

Problema 1. (4 puncte)

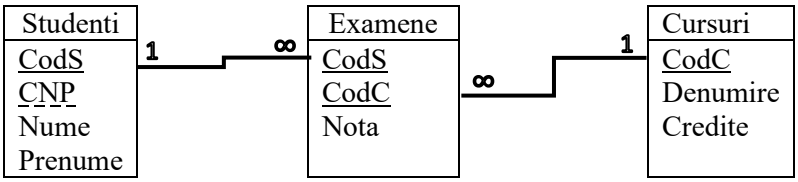
Un institut de cercetare stochează informații despre ierbarele utilizate în activitățile proprii de cercetare. Ierbarele sunt depozitate în diferite spații de depozitare din clădirea institutului și sunt utilizate de către angajații institutului de cercetare. Pentru a gestiona informațiile despre angajați, spații de depozitare, ierbare, plante și zone geografice se folosește o bază de date relațională:

- Un angajat al institutului de cercetare are un cod, un nume, o adresă de email (unică pentru fiecare angajat), data nașterii, funcția pe care o ocupă în cadrul institutului de cercetare și salariul.
- Un spațiu de depozitare are un cod, o denumire, o descriere și un angajat responsabil de administrarea sa. Fiecare angajat poate fi responsabil cu administrarea mai multor spații de depozitare, dar un spațiu de depozitare poate fi administrat de către un singur angajat.
- O regiune geografică are un cod, o denumire, o descriere și o țară.
- Un ierbar are un cod, un titlu, o descriere, o dată la care a fost preluat în gestiunea institutului de cercetare și este depozitat într-un anumit spațiu de depozitare din cadrul institutului. Fiecare spațiu de depozitare poate stoca mai multe ierbare, dar un ierbar este stocat într-un singur spațiu de depozitare.
- O plantă are un cod, o denumire științifică, o denumire comună, clasa, ordinul și o valoare care indică dacă este protejată sau nu. Un ierbar poate conține mai multe plante, iar o plantă poate apărea în mai multe ierbare. Fiecare plantă poate fi asociată fiecărui ierbar o singură dată. Pentru fiecare pereche formată dintr-o plantă și un ierbar (de exemplu perechea formată din planta p1 și ierbarul i1) se va stoca și data la care a fost adăugată planta în ierbar, împreună cu regiunea geografică din care a fost recoltată planta.

Realizați o schemă relațională BCNF pentru baza de date, evidențiind riguros cheile primare, cheile candidat și cheile externe. Realizați schema într-una din manierele indicate în exemplul de mai jos:

* exemplu pentru tabelele Studenti, Cursuri și Examene:

V1. Diagramă cu tabele, chei primare subliniate cu linie continuă, chei candidat subliniate cu linie întreruptă, legături trasate direct între cheile externe și cheile primare / candidat corespunzătoare (de exemplu, legătură trasată între coloana CodS din Examene și coloana CodS din Studenti).



V2.
Studenti[CodS, CNP, Nume, Prenume]
Cursuri[CodC, Denumire, Credite]
Examene[CodS, CodC, Nota]
Cheile primare sunt subliniate cu linie continuă, iar cheile candidat sunt subliniate cu linie întreruptă. {CodS} este cheie externă în Examene și face referire la {CodS} din Studenti. {CodC} este cheie externă în Examene și face referire la {CodC} din Cursuri.

Problema 2. (5 puncte)

Considerăm următoarele relații dintr-o bază de date care stochează informații despre operele de artă și expozițiile temporare ale unui muzeu:

Expozitii[CodExpozitie, Titlu, Descriere, DataVernisajului, NumarSala]
PerioadeArtistice[CodPerioada, Denumire]

Artisti[CodArtist, Nume, DataNasterii, *CodPerioada*]
OpereDeArta[CodOperaDeArta, Titlu, AnulCrearii, *CodArtist, CodExpozitie*, TipOperaDeArta]
Cheile primare sunt subliniate. Cheile externe sunt scrise cursiv și au aceeași denumire cu coloanele la care fac referire.

a. Scrieți o interogare SQL care returnează toate expozițiile (titlul expoziției, data vernisajului, numărul sălii) care conțin cel puțin o operă de artă realizată de artistul cu numele „Pablo Picasso", dar nu includ nicio operă de artă realizată de artistul cu numele „Claude Monet". Eliminați duplicatele.

b. Se dau instanțele următoare ale relațiilor PerioadeArtistice, Artisti și OpereDeArta:

PerioadeArtistice:

Cod Perioada	Denumire
1	Cubism
2	Impresionism
3	Postimpresionism

Artisti:

Cod Artist	Nume	DataNasterii	Cod Perioada
1	Pablo Picasso	25.10.1881	1
2	Claude Monet	14.11.1840	2
3	Vincent van Gogh	30.03.1853	3
4	Edgar Degas	19.07.1834	2

OpereDeArta:

CodOpera DeArta	Titlu	Anul Crearii	Cod Artist	Cod Expozitie	TipOpera DeArta
1	Guernica	1937	1	1	Pictura
2	Jacqueline with Flowers	1954	1	1	Pictura
3	Water lilies	1919	2	2	Pictura
4	The Little Fourteen-Year-Old Dancer	1880	4	2	Sculptura
5	The Ballet Class	1874	4	2	Pictura
6	Sunflowers	1888	3	3	Pictura
7	The Starry Night	1889	3	2	Pictura
8	Cafe Terrace at Night	1888	3	1	Pictura

b1. Precizați rezultatul evaluării interogării de mai jos pe instanțele date. Menționați strict valorile tuplului / tuplurilor și denumirile coloanelor din rezultat, fără a prezenta toți pașii evaluării interogării.

```
SELECT A.Nume, PA.Denumire, COUNT (O.CodOperaDeArta) as Nr
FROM Artisti A
        INNER JOIN PerioadeArtistice PA ON A.CodPerioada = PA.CodPerioada
        INNER JOIN OpereDeArta O ON O.CodArtist = A.CodArtist
GROUP BY A.CodArtist, A.Nume, PA.Denumire
HAVING COUNT(O.CodOperaDeArta) >=
  (SELECT AVG(T.Nr)
   FROM          (SELECT A1.CodArtist, COUNT(*) AS Nr
                  FROM Artisti A1
                        INNER JOIN PerioadeArtistice PA1 ON A1.CodPerioada = PA1.CodPerioada
                        INNER JOIN OpereDeArta O1 ON O1.CodArtist = A1.CodArtist
                  GROUP BY A1.CodArtist
                        ) T
          )
```

b2. Explicați dacă următoarele dependențe funcționale sunt satisfăcute sau nu de datele din instanța OpereDeArta:

- { CodOperaDeArta } → { CodArtist }
- { CodExpozitie } → { CodOperaDeArta }.

SUBIECT Sisteme de operare

Problema 1. (5 puncte) Răspundeți la următoarele întrebări despre execuția programului `./a.out` compilat din codul sursă de mai jos, considerând că toate include-urile necesare sunt prezente și că apelurile sistem `fork` și `write` se execută cu succes.

```
 1  int main() {
 2    int i;
 3    char buf[3] = {'L','0','\n'};
 4    for (i = 0; i < 3; i++) {
 5      buf[1] = '0' + i;
 6      write(i%2+1, buf, 3);
 7      fork();
 8    }
 9    write(1, "END\n", 4);
10    for (i = 0; i < 3; i++)
11      wait(NULL);
12    return 0;
13  }
```

a) De câte ori se afișează linia `END` la execuția `./a.out`? Justificați răspunsul.

b) De câte ori va apărea în fișierul `out` fiecare dintre liniile `L0`, `L1` și `L2`, în urma execuției `./a.out > out`? Justificați răspunsul.

c) Câte procese se creează (în afara procesului inițial) în total în timpul execuției? Justificați răspunsul.

d) Explicați în detaliu funcționarea buclei de la liniile `10-11` și precizați câți fii direcți așteaptă procesul inițial. Justificați răspunsul.

e) De câte ori se afișează în total liniile `L (L0+L1+L2)`, la execuția `./a.out`, dacă linia `6` este mutată imediat după linia `7`? Justificați răspunsul.

Problema 2. (4 puncte) Răspundeți la următoarele întrebări despre execuția scriptului Shell UNIX `./a.sh` de mai jos, considerând că acesta folosește fișierul `f.txt` dat. Notă: numerotarea liniilor NU aparține de conținutul fișierelor.

```
1  #!/bin/bash
2  while read NAME GRADE; do
3    if [ $GRADE -ge 5 ]; then
4      echo "$NAME"
5    fi
6  done < f.txt | sort | uniq -c | sort -nr | head -n 1
```

f.txt

```
1  Anca 7
2  Radu 9
3  Anca 8
4  Radu 2
5  Anca 3
```

a) Ce afișează doar bucla `while` (liniile `2-6`, înainte de primul `| sort`)? Justificați răspunsul.
b) Care este efectul expresiei `sort | uniq -c`? Justificați răspunsul.
c) Ce afișează scriptul complet? Justificați răspunsul.
d) Cum se modifică rezultatul dacă se elimină comanda `sort` dinaintea lui `uniq -c`? Justificați răspunsul.

BAREM INFORMATICĂ
VARIANTA 1

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1. – 3p**

Funcția de bază – **0.4p** din care
- signatura – 0.1p
- implementare – 0.3p

Funcția auxiliară – **2.6p** din care
- signatura – 0.1p
- implementare iterativă funcție auxiliară având complexitate timp $O(\log_2 n)$ – 2.5p
  * soluție recursivă având complexitate timp $O(\log_2 n)$ – 1p
  * soluție iterativă având complexitate timp $O(n)$ – 0.5p

Cerința **2. – 1p**
- caz favorabil **0.3p** din care
  - complexitate – 0.15
  - justificare – 0.15
- caz mediu **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2
- caz defavorabil **0.3p** din care
  - complexitate – 0.15
  - justificare – 0.15

Cerința **3.a) – 1.7p**

Definirea clasei Organisation – **0.2p** din care
- atribut – 0.1
- metode **buildOrganisation**, **getUnit** – 0.1

Definirea clasei Unit – **0.5p** din care
- atribute – 0.2
- constructor – 0.1
- metode **getId, getHead, getUnits, addUnit** – 0.2

Definirea clasei Compartment– **0.7p** din care
- relația de moștenire – 0.2
- atribut – 0.1
- **constructor (a1)** – 0.3
- metode **getEmployees, addEmployee** – 0.1

Definirea clasei Employee – **0.3p** din care
- atribute– 0.2
- metoda **getSalary** – 0.1

Funcția **3.b)** – **2p**
- signatura – **0.1p**
- implementare funcție – **1.8p**
- returnare rezultat – **0.1p**

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

Funcția principală **3.d)** – **0.3p**
- construire obiect **o** – 0.1p
- apel metoda **buildOrganisation**() – 0.1p
- apel funcții b) și c) – 0.1p

BAREM INFORMATICĂ
VARIANTA 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:

Nume	**Denumire**	**Nr**
Pablo Picasso	Cubism	2
Vincent van Gogh	Postimpresionism	3
Edgar Degas	Impresionism	2

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

- **b2 -** { CodOperaDeArta } → { CodArtist } este satisfăcută – **0.25p**; **0.25p** explicație
  - { CodExpozitie } → { CodOperaDeArta } 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 1

Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) 8 = 2^3 procese

1p – b) L0 o dată, L1 niciodată, L2 de patru ori

1p – c) 7 procese fii, 2^3 fără procesul inițial

1p – d) se așteaptă 3 procese fiu, dacă acestea nu există, wait returnează -1; procesul inițial așteaptă 3 procese copil

1p – e) Liniile L se afișează în total de 14 ori (L0×2 + L1×4 + L2×8)

Problema 2 (4p)

1p – a) Anca, Radu, Anca (în această ordine, câte una pe linie); numele doar pentru liniile cu nota >=5

1p – b) `sort` sortează în odine alfabetică liniile primite, iar `uniq` elimină liniile consecutive identice și afișează de câte ori apare fiecare, folosind opțiunea `-c`

[opțional: Aici obținem: 2 Anca şi 1 Radu (numărul precedă numele);]

1p – c) 2 Anca - `sort -nr` ordonează descrescător numeric după numărul din față, `head -n 1` păstrează prima linie

1p – d) Neavând linii consecutive identice, uniq -c produce 1 Anca, 1 Radu, 1 Anca, rezultatul final fiind 1 Anca