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

# Proba scrisă a examenului de licenţă, 3 septembrie 2024
# Informatică Română

# VARIANTA 2

## NOTĂ

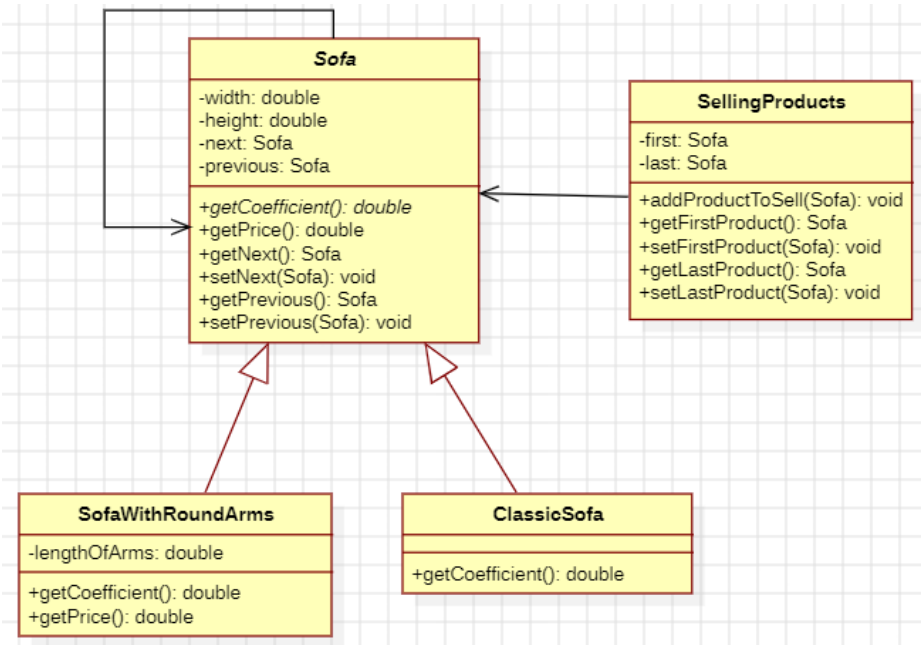
- Toate subiectele sunt obligatorii. La toate subiectele se cer rezolvări cu soluții complete.
- Nota minimă ce asigură promovarea este 5.00.
- Timpul efectiv de lucru este de 3 ore.

## SUBIECT Algoritmică şi 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 și căutare predefinite.**

*Pentru **tipurile de date** puteți folosi biblioteci existente (C++, Java, C#).*

a) Se dă următoarea diagramă UML conținând clasele ***Sofa*** (Canapea), **SofaWithRoundArms** (CanapeaCuBrațeRotunde), **ClassicSofa** (CanapeaClasică), **SellingProducts** (ProduseDeVânzare, memorează canapelele de vânzare). Constructorii nu sunt indicați pe diagramă.




- Atributele *width* și *height* din clasa ***Sofa*** (lățimea și înălțimea) și atributul *lengthOfArms* din clasa **SofaWithRoundArms** (lungimea brațelor măsurată în metri) trebuie să fie valori strict pozitive. Constructorii trebuie să impună constrângerile.
- Clasa abstractă ***Sofa*** are o metoda abstractă *getCoefficient()* care returnează un coeficient care se aplică pentru calculul prețului unei canapele.

- Coeficientul aplicat pentru calculul prețului pentru un obiect de tip **ClassicSofa** este 1.5, iar pentru un obiect de tip **SofaWithRoundArms** coeficientul este 2.
- Prețul unei canapele este calculat ca fiind coeficientul înmulțit cu suma între *width* și *height*. Pentru **SofaWithRoundArms**, la acest preț se adaugă prețul pentru brațe. Un metru de brațe costă 10 lei.
- Metoda *addProductToSell(Sofa)* din clasa **SellingProducts** adaugă obiectul dat ca parametru în secvența de canapele deja existentă**.** Adăugarea se va face în așa fel încât afișarea secvenței de canapele în ordinea descrescătoare a prețului să fie efectuată în timp liniar.

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

a1) Declarați toate clasele, atributele și metodele conform diagramei de mai sus.

Implementați doar următoarele metode:

a2) Metoda *getPrice()* din clasa **SofaWithRoundArms.**
a3) Constructorii claselor ***Sofa,*** **SofaWithRoundArms** și **SellingProducts**.
a4) Metoda *addProductToSell(Sofa)* din clasa **SellingProducts**.
a5) Metoda *getCoefficient()* din clasa **ClassicSofa** și metoda *setPrevious(Sofa)* din clasa ***Sofa***.

b) Definiți o funcție care primește ca parametru un obiect *s* de tip **SellingProducts** și afișează, în timp liniar, prețurile canapelelor din *s*, în ordine crescătoare. Implementările care nu se încadrează în clasa de complexitate indicată se punctează parțial.
c) Definiți o funcție care primește ca parametru un obiect *s* de tip **SellingProducts,** două valori reale *startPrice* și *endPrice*, și elimină din *s* acele canapele al căror preț este în intervalul [*startPrice*, *endPrice*]. Se va folosi o implementare cu o complexitate timp $O(n)$ ($n$ este numărul de canapele din lista asociată obiectului **SellingProducts**). Implementările care nu se încadrează în clasa de complexitate indicată se punctează parțial.
d) Construiți un obiect de tip **SellingProducts** în care se adaugă 4 canapele (alegeți valori pentru proprietățile neprecizate)**:** două de tip **ClassicSofa** și două de tip **SofaWithRoundArms** având 1.2 și respectiv 0.9 metri de brațe**.** Apelați funcția de la punctul b) și apoi funcția de la c) (alegeți valori pentru *startPrice* si *endPrice*).
e) Precizați complexitatea timp a metodei *addProductToSell(Sofa)* din clasa **SellingProducts,** în cazurile *favorabil*, *mediu* și *defavorabil*.

**SUBIECT Baze de date**

**Problema 1. (4 puncte)**

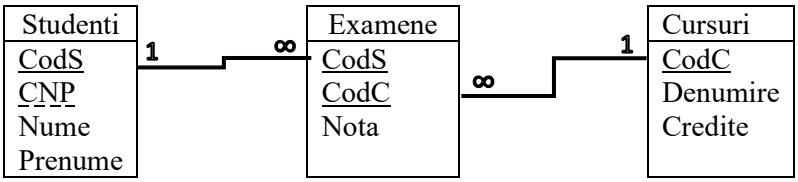
Organizatorii unor competiții de cules scoici stochează următoarele informații într-o bază de date relațională:

- Un participant are un cod, un nume, o adresă de email și o țară de proveniență.
- O plajă are un cod, un nume, o descriere și o localitate.
- O competiție are un cod, o denumire, o dată la care este programată, o oră de început, o oră de sfârșit și o plajă asociată. Pe o plajă se pot desfășura mai multe competiții.
- Un tip de scoici are un cod, o denumire și o descriere.
- În cadrul fiecărei competiții, participanții culeg scoici de diferite tipuri de pe plaja unde are loc competiția. În baza de date se stochează doar numărul total de scoici culese de fiecare participant în fiecare competiție, pentru fiecare tip de scoici (de exemplu, participantul *p* a cules: 10 scoici de tipul *t1* și 15 scoici de tipul *t2* în competiția *c1*; 50 de scoici de tipul *t1*, 15 scoici de tipul *t2* și 20 de scoici de tipul *t3* în competiția *c2* etc).

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 despre o grădiniță particulară:
Grupe[CodGrupa, NumeGrupa, NrSala]
Parinti[CodParinte, NumeParinte, Adresa, NrTelefon]
Copii[CodCopil, NumeCopil, *CodParinte,* DataNasterii, *CodGrupa*]
PlatiTaxe[CodPlata, *CodCopil*, DataPlatii, SumaPlatita]

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, pentru fiecare grupă pentru care s-a plătit cel puțin o taxă: codul, numele grupei și valoarea totală a taxelor plătite pentru copiii din grupă.

**b.** Se dau instanțele următoare ale relațiilor Grupe, Parinti și Copii:

**Grupe:**

| Cod Grupa | Nume Grupa | NrSala |
|---|---|---|
| 1 | Buburuze | 1 |
| 2 | Veverițe | 5 |
| 3 | Albinuțe | 8 |

**Parinti:**

| Cod Parinte | Nume Parinte | Adresa | NrTelefon |
|---|---|---|---|
| 1 | P1 | A1 | 1111111111 |
| 2 | P2 | A2 | 2222222222 |
| 3 | P3 | A3 | 3333333333 |

**Copii:**

| Cod Copil | Nume Copil | Cod Parinte | DataNasterii | Cod Grupa |
|---|---|---|---|---|
| 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.** 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 p.NumeParinte, p.Adresa
FROM Parinti p
    INNER JOIN Copii c ON p.CodParinte = c.CodParinte
WHERE c.CodGrupa = (SELECT g.CodGrupa
                    FROM Grupe g
                    WHERE g.NumeGrupa = 'Buburuze')
INTERSECT
SELECT p.NumeParinte, p.Adresa
FROM Parinti p
    INNER JOIN Copii c ON p.CodParinte = c.CodParinte
GROUP BY p.CodParinte, p.NumeParinte, p.Adresa
HAVING COUNT(DISTINCT c.CodGrupa) = 1
```

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

- {CodCopil} → {CodParinte}
- {CodParinte} → {CodGrupa}.

**SUBIECT Sisteme de operare**

**Problema 1 (5 puncte).** Răspundeţi la următoarele întrebări despre execuţia programului `./a.out` compilat din codul sursă de mai jos, considerând că toate include-urile necesare sunt prezente, că apelul sistem `fork` şi apelul sistem `execlp` se execută cu succes şi FIFO-urile `fa` şi `fb` au fost create în prealabil. Comanda `bash` cu opţiunea `-c` execută comanda din stringul dat ca valoare opţiunii `-c`.

```
 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) Ce va afişa execuţia de mai jos? Justificaţi răspunsul.

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

b) Cum e afectată execuţia dacă blocul IF de pe liniile 8-16 este mutat între liniile 3 şi 4? Justificaţi răspunsul.

c) Ce va afişa execuţia de mai jos, dacă în linia 20 cifra `1` este schimbată în `2`? Justificaţi răspunsul.

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

d) Cum va fi afectată execuţia dacă linia 14 este comentată? Justificaţi răspunsul.

e) Cum va fi afectată execuţia dacă liniile 4, 6 şi 7 sunt comentate? Justificaţi răspunsul.

**Problema 2 (4 puncte).** Răspundeţi la următoarele întrebări despre execuţia scriptului Shell UNIX `./a.sh` de mai jos.

```
 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) Ce va afişa execuţia de mai jos?

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

b) Care vor fi valorile variabilelor `A`, `B` şi `C` la fiecare iteraţie a ciclului `while` şi care va fi valoarea afişată de execuţia de mai jos?

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

c) Care vor fi valorile variabilelor `A`, `B` şi `C` la fiecare iteraţie a ciclului `while` şi care va fi valoarea afişată de linia de comandă de mai jos? Justificaţi răspunsul.

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

d) Explicaţi în detaliu funcţionarea liniilor 17-19

# 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