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

Proba scrisă a examenului de licenţă, 1 iulie 2024
Informatică Română

VARIANTA 1

NOTĂ

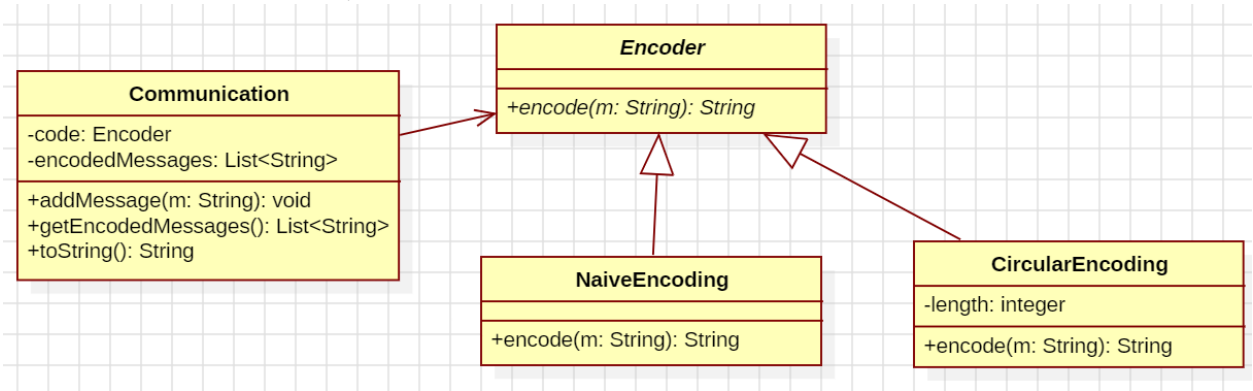
- Toate subiectele sunt obligatorii. La toate subiectele se cer rezolvări cu soluții complete.
- Nota minimă ce asigură promovarea este 5.00.
- Timpul efectiv de lucru este de 3 ore.

SUBIECT Algoritmică şi programare

- **Se va indica limbajul de programare folosit.**
- **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 **Communication** (Comunicare), **CircularEncoding** (CodificareCirculară), **NaiveEncoding** (CodificareNaivă), **Encoder** (Codificator). Constructorii nu sunt indicați pe diagramă.



- Clasa abstractă ***Encoder*** are o metodă abstractă *encode*(*m:String*) care codifică mesajul *m*.
- Atributul *length* din clasa **CircularEncoding** trebuie să fie o valoare pozitivă, strict mai mare decât 2. Atributul *code* din clasa **Communication** trebuie să fie nenul. Constructorii trebuie să impună constrângerile.
- Metoda **encode(m: String)** din clasa **CircularEncoding** permută circular mesajul **m** cu *length* poziții spre stânga (de exemplu, mesajul 'ENCODER' permutat cu 2 poziții spre stânga este 'CODEREN'). Metoda **encode(m: String)** din clasa **NaiveEncoding** returnează oglinditul mesajului **m**. Prin oglinditul unui șir de caractere se înțelege șirul de caractere în ordine inversă (de exemplu, oglinditul șirului de caractere 'MESAJ' este 'JASEM').
- Clasa **Communication** memorează o listă *encodedMessages* de mesaje codificate conform metodei **encode** a atributului *code*. Metoda **addMessage(m: String)** din clasa **Communication** adaugă mesajul **m** codificat în lista *encodedMessages*, iar metoda **getEncodedMessages()** returnează lista *encodedMessages*. Metoda **toString()** din clasa **Communication** returnează șirul

de caractere obținut concatenând toate mesajele din lista *encodedMesages*, separate prin caracterul '#'.

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) Constructorii claselor **CircularEncoding** și **Communication**.
a3) Metoda **toString():String** din clasa **Communication.**

b) Definiți o funcție care primește ca parametri două liste **L1** și **L2** de mesaje (de tip string) ordonate alfabetic și construiește, prin interclasare, o a treia listă (ordonată alfabetic), conținând toate mesajele din **L1** și **L2**. Lista construită prin interclasare va fi returnată ca rezultat al funcției. Se va folosi o implementare recursivă. Implementarea iterativă se punctează parțial.
c) Definiți o funcție care primește ca parametri două mesaje **m1** și **m2** de tip **String** și verifică (folosind spațiu suplimentar de memorare $\theta(1)$) dacă unul dintre mesaje este o permutare circulară a celuilalt. Implementările al căror spațiu suplimentar de memorare nu se încadrează în clasa de complexitate indicată se punctează parțial.
d) Definiți o funcție care primește ca parametru un obiect *c* de tip **Communication** și afișează numărul de mesaje distincte din lista *encodedMessages* asociată obiectului *c*. Se cere o implementare având o complexitate timp în caz mediu $\theta(n)$ (n fiind numărul de mesaje din lista *encodedMessages*, $n \leq 10^6$). Se consideră că toate mesajele din lista *encodedMessages* au cel mult 10 caractere. Soluțiile care nu se încadrează în complexitatea cerută se punctează parțial.
e) Construiți două obiecte **o1** și **o2** de tip **Communication** în care se adaugă, în această ordine, mesajele "bac", "cod" și "apt"**: o1** folosește o codificare de tip **CircularEncoding** cu lungimea *length* egală cu 10, **o2** folosește o codificare de tip **NaiveEncoding.** Folosind funcția de la punctul b), construiți lista ordonată alfabetic obținută interclasând mesajele codificate conținute în **o1** și **o2** și apoi afișați lista obținută.
f) Precizați complexitatea timp a funcției de la b).
g) Precizați complexitatea timp în cazurile *favorabil*, *mediu* și *defavorabil* pentru operația de adăugare într-o tabelă de dispersie a unui element a cărui cheie este număr natural.

SUBIECT Baze de date

Problema 1. (4 puncte)

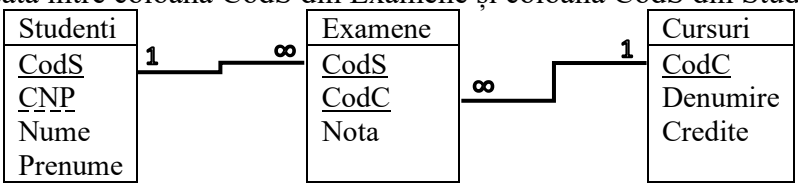
Un lanț de cafenele folosește o bază de date relațională pentru gestiunea datelor referitoare la cafenele, preparate, clienți și comenzi.

- O cafenea are un cod, o adresă (poate fi un atribut simplu – șir de caractere), capacitate (număr locuri) și număr de telefon (unic pentru fiecare cafenea). O cafenea: are sau nu terasă, oferă sau nu Wi-Fi.
- Lanțul de cafenele oferă aceeași gamă de preparate în toate cafenelele, deci nu se stochează asocieri între cafenele și preparate. Un preparat are un cod, denumire, descriere și același preț în toate cafenelele.
- Clienții își pot instala o aplicație pe telefon pentru realizarea comenzilor. Pentru fiecare astfel de client, se stochează în baza de date codul clientului și numărul său de telefon (unic pentru fiecare client); nu se stochează alte date cu caracter personal referitoare la client (nume, prenume etc), aplicație sau instalarea aplicației.
- O comandă are un cod și este făcută de un client într-o cafenea, într-o anumită zi (dată calendaristică). Comanda poate include mai multe preparate. Pentru fiecare preparat din comandă, se reține numărul total de bucăți (sau porții) comandate. Exemplu: o comandă pentru 2 cafele *Espresso*, 3 *Cappuccino* și 2 cafele *Americano*. Un client poate face mai multe comenzi în aceeași zi, în aceeași cafenea. Un client poate face mai multe comenzi în aceeași zi, în cafenele diferite. Un client poate face mai multe comenzi în aceeași cafenea, în zile diferite. Într-o cafenea pot face comenzi, bineînțeles, mai mulți clienți (în aceeași zi sau în zile diferite).

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)

Se dau următoarele scheme relaționale:

Regizori[CodRegizor, Nume, Tara]
Filme[CodFilm, Titlu, An, Incasari, *CodRegizor*]
Premii[CodPremiu, Denumire, Institutie]
FilmePremii[*CodFilm*, *CodPremiu*, Categorie, An]
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 codul și numele fiecărui regizor care a regizat cel puțin un film care a câștigat cel puțin un premiu cu denumirea *Oscar* și cel puțin un premiu cu denumirea *Ursul de Aur*.

b. Se dau instanțele următoare pentru relațiile Filme, Premii și FilmePremii:

Filme

Cod Film	Titlu	An	Incasari	Cod Regizor
1	t1	1990	1.10	1
2	t2	2023	1.50	2
3	t3	2020	5.90	1
4	t4	1965	2.00	1
5	t5	2000	4.50	2

FilmePremii

Cod Film	Cod Premiu	Categorie	An
1	1	1	1990
1	2	3	1991
2	1	1	2023
5	1	1	2000
5	2	4	2000

Premii

Cod Premiu	Denumire	Institutie
1	p1	i1
2	p2	i2

Notă: Incasari în Filme se exprimă în număr de milioane de lei.

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 f.CodFilm, f.Titlu
FROM Filme f
WHERE NOT EXISTS
        (SELECT p.CodPremiu
        FROM Premii p
        EXCEPT
        SELECT DISTINCT fp.CodPremiu
        FROM FilmePremii fp
        WHERE fp.CodFilm = f.CodFilm)
        AND f.Incasari > (SELECT MAX(IncMed)
                        FROM (SELECT AVG(f2.Incasari) IncMed
                                FROM Filme f2
                                GROUP BY f2.CodRegizor) t
                        )
```

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

- $\{An\} \rightarrow \{CodRegizor\}$
- $\{CodRegizor\} \rightarrow \{Titlu\}$

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 şi că apelul sistem `fork` şi apelul sistem `execlp` se execută cu succes.

```
 1  int main(int argc, char** argv) {
 2    int n;
 3    char s[10];
 4
 5    n = atoi(argv[1]);
 6    if(n > 0) {
 7      fork();
 8      if(wait(NULL) > 0) {
 9        sprintf(s, "%d", n-1);
10        execlp(argv[0], argv[0], s, NULL);
11      }
12      printf("%d\n", n);
13    }
14    return 0;
15  }
```

a) Explicaţi în detaliu funcţionarea liniei 10.

b) Câte procese creează execuţia de mai jos, excluzând procesul părinte iniţial? Justificaţi răspunsul.

```
./a.out 3
```

c) Ce va afişa procesul părinte în execuţia de mai jos? Justificaţi răspunsul.

```
./a.out 1
```

d) Specificaţi liniile, în ordinea apariţiei, afişate de execuţia de mai jos. Justificaţi răspunsul.

```
./a.out 3
```

e) Specificaţi liniile, în ordinea apariţiei, afişate de execuţia de mai jos, dacă pe linia 8 caracterul `'>'` se înlocuieşte cu `'<'`. Justificaţi răspunsul.

```
./a.out 3
```

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

```
1  #!/bin/bash
2
3  while read X; do
4    if echo "$X" | grep -E -q "^[^+-]+([+-][^+-]+)+$"; then
5      echo "$X" | sed -E "s/[^0-9+-]//g" | sed -E "s/([+-])/ \1 /g"
6    fi
7  done < a.txt | while read A B C; do
8    expr $A $B $C
9  done
```

a.txt

```
1  xx2x+x4xx
2  2 + 1 0 * 3
3  2 + 4 - 5
```

a) Explicaţi în detaliu funcţionarea liniei 5 din script.
b) Specificaţi valorile variabilelor `A`, `B`, `C` şi rezultatul afişat de execuţia scriptului pentru linia 1 a fişierului `a.txt`.
c) Specificaţi valorile variabilelor `A`, `B`, `C` şi rezultatul afişat de execuţia scriptului pentru linia 2 a fişierului `a.txt`.
d) Specificaţi valorile variabilelor `A`, `B`, `C` şi rezultatul afişat de execuţia scriptului pentru linia 3 a fişierului `a.txt`.

BAREM INFORMATICĂ
VARIANTA 1

Subiect Algoritmică şi Programare

Oficiu – 1p
Cerința **a) – 2p**
- Definirea clasei Encoder – **0.3p** din care
  - clasă abstractă – 0.1
  - metoda **encode** – 0.2
- Definirea clasei CircularEncoding– **0.55p** din care
  - relația de moștenire – 0.25
  - atribut – 0.1
  - **constructor (a2)** – 0.1
  - metoda **encode** – 0.1
- Definirea clasei NaiveEncoding – **0.35p** din care
  - relația de moștenire – 0.25
  - metoda **encode**– 0.1
- Definirea clasei Communication – **0.8p** din care
  - atribute– 0.2
  - **constructor (a2)** - 0.2
  - metoda **addMessage** – 0.1p
  - metoda **toString (a3)** – 0.3p

Funcția **b)** – **2.5p**
- signatura – **0.1p**
- implementare recursivă interclasare – **2.3p**
  - * soluție iterativă corectă – 0.75p
- returnare rezultat – **0.1p**

Funcția **c)** – **1p**
- signatura – **0.1p**
- implementare folosind spațiu suplimentar de memorare $\theta(1)$– **0.8p**
  - * soluție *out-of-place* – 0.25p
- returnare rezultat – **0.1p**

Funcția **d)** – **1p**
- signatura – **0.1p**
- implementare având complexitate timp în caz mediu $\theta(n)$ – **0.8p**
  - * soluție având complexitate timp în caz mediu $\theta(n\log_2 n)$ – 0.5p
  - * soluție având complexitate timp în caz mediu $\theta(n^2)$ – 0.25p
- returnare rezultat – **0.1p**

Funcția principală **e)** – **0.5p**
- construire obiecte **o1** și **o2** – 0.1p
- adăugare mesaje în **o1** și **o2** - 0.1p
- apel funcție b) – 0.1p
- afișare lista rezultată prin interclasare - 0.2 p

Cerința **f)** – **1p**
Cerința **g)** – **1p**
- favorabil (**0.25p**)
- mediu (**0.5p**)
- defavorabil (**0.25p**)

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:

CodFilm	Titlu
5	t5

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

- **b2** - {An}→{CodRegizor} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodRegizor}→{Titlu} 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 1

Subiect: Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) execlp lansează în execuţie programul ./a.out, înlocuind segmentul de cod al procesului curent.

1p – b) trei procese noi, unul pentru argumentul 3, unul pentru argumentul 2 şi unul pentru argumentul 1

1p – c) nimic, pentru că având un proces fiu, procesul părinte intră în IF, execută exec şi nu mai ajunge la printf

1p – d) 3, 2, 1 pentru că părintele nu execută execlp (şi nu creează un nou proces fiu) până când cel curent nu îşi încheie execuţia

1p – e) 1, 2, 3 pentru că părintele afişează doar după încheiere procesului fiu

Problema 2 (4p)

1p – a) se elimină din valoarea variabilei X toate caracterele care nu sunt +, - sau cifră, apoi se adaugă câte un spaţiu în jurul operatorilor + şi – şi rezultatul obţinut se scrie la ieşirea standard

1p – b) A='2' B='+' C='4' REZULTAT=’6'

1p – c) A='2' B='+' C='103' REZULTAT=’105'

1p – d) A='2' B='+' C='4 - 5' REZULTAT=’1'