UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Proba scrisă a examenului de licență, 5 septembrie 2023
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 și programare

Notă

- 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.
- În Pseudocod se pot utiliza exclusiv următoarele instrucțiuni: *atribuire*, *pentru*, *dacă*, *cât timp*, *repetă*, *afișare* și *returnare*.
- 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.

Problema 1. (1.5 puncte)

Fie șirul $a[1], a[2]....a[n]$ ($n \geq 1$) de numere naturale distincte în ordine descrescătoare și un număr natural x. Scrieți un subalgoritm în Pseudocod, având complexitatea timp $O(\log_2 n)$, care returnează poziția *poz* pe care apare x în șirul a (în cazul în care x apare în șir) sau poziția pe care ar trebui inserat x în șirul a astfel încât să se păstreze ordinea descrescătoare a elementelor. Justificați complexitatea. **Observație.** Inserarea unui element pe o poziție k în șir presupune deplasarea elementelor de pe pozițiile k, $k+1$, ..., n cu o poziție spre dreapta și incrementarea lungimii șirului.

Exemplu. Dacă $n=4$, șirul a este (14, 12, 9, 5) atunci:

- pentru $x=15$ se va returna $poz=1$, deoarece șirul obținut prin inserarea lui 15 ar fi (**15**, 14, 12, 9, 5);
- pentru $x=11$ se va returna $poz=3$, deoarece șirul obținut prin inserarea lui 11 ar fi (14, 12, **11**, 9, 5);
- pentru $x=12$ se va returna $poz=2$, deoarece elementul **12** apare deja în șir pe poziția 2.

Problema 2. (1.5 puncte)

Scrieți un subalgoritm în Pseudocod care, având ca parametri de intrare un text reprezentat sub forma unui șir de caractere $x[1], x[2]....x[n]$ ($n \geq 1$) și un număr întreg k, $0 < k \leq n$, determină șirul de caractere $y[1], y[2]....y[n]$ reprezentând permutarea circulară a șirului x cu k poziții spre dreapta. Precizați complexitatea timp a subalgoritmului.

Exemplu. Dacă $n=6$ și textul este S, T, R, I, N, G, atunci pentru $k=2$ șirul y va fi N, G, S, T, R, I.

Problema 3. (1.5 puncte)

Se dă un șir a cu n elemente ($a[1]$, ..., $a[n]$), care reprezintă o permutare a elementelor $\{1, ..., n\}$ ($n>0$). Scrieți un algoritm în Pseudocod, având complexitatea timp în caz defavorabil $\theta(n)$, care determină dacă șirul a poate reprezenta parcurgerea în inordine a unui arbore binar de căutare cu n noduri etichetate de la 1 la n. În caz afirmativ, să se afișeze valoarea 0, în caz contrar să se afișeze o listă de interschimbări prin care se poate obține un șir care respectă condiția. Orice soluție care nu se încadrează în clasa de complexități indicată va primi punctaj parțial.

Exemplu. Dacă $n = 3$, șirul a este (3, 2, 1), atunci se afișează interschimbarea 1 3 (trebuie interschimbate elementele de pe pozițiile 1 și 3 - astfel un posibil arbore a cărui parcurgere în inordine este (1, 2, 3) are rădăcina 2, fiul stâng 1 și fiul drept 3).

Problema 4. (2.25 puncte)

Se dă următorul program (incomplet) în C++. Completați implementarea claselor **Computer** și **SorterByTotalPrice** astfel încât funcția **main** să afișeze pe ecran "HC85, 165". Metoda **sort** din clasa **SorterByTotalPrice** va primi o listă de produse și le va sorta descrescător după prețul total. Sortarea se va face prin apelul funcției **std::sort** din biblioteca standard C++. Prețul total al unui calculator se calculează prin adunarea unei taxe la prețul de bază.

```
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>

class Product
{
   private:
       std::string name;
       int baseprice;

   public:
       Product(const std::string& name, int baseprice):
                     name{name},
                     baseprice{baseprice} {}
       virtual int total_price() {
              return this->baseprice;
       }
       virtual std::string toString() {
              return this->name;
       }
      virtual ~Product() {}
};

class Computer: public Product {
   private:
       int tax;
       //A: missing code
};

class SorterByTotalPrice {
       //B: missing code
};
int main()
{
       std::vector<Product*> computers{new Computer{"HC90", 140, 10},
                                       new Computer{"HC91", 100, 12},
                                       new Computer{"HC85", 150, 15}};
       SorterByTotalPrice::sort(computers);
       std::cout << computers[0]->toString();
       for (auto c: computers)
           delete c;
       return 0;
}
```

Problema 5. (2.25 puncte)

Implementați în limbajul C++ clasele de la cerințele 1-3 și apoi funcția de la cerința 4.

1. Clasa ***Contact*** este abstractă, avand un atribut privat *name* de tip string, o metoda abstractă (pur virtuală) **sendMessage**, care primește un parametru *message* de tip string și o metoda **getName**, care va returna valoarea atributului *name*.
2. Clasa **Person** extinde clasa **Contact**, mai are un atribut privat *number* de tip string, iar metoda **sendMessage** primește ca parametru un mesaj de tip string și tipărește numele și numărul persoanei, precum și mesajul.
3. Clasa **Group** extinde clasa **Contact** și poate să conțină zero sau mai multe contacte. Metoda **addContact** primește ca parametru un contact pe care îl adaugă în lista de contacte. Metoda **sendMessage** tipărește numele și numerele tuturor contactelor din lista de contacte, precum și mesajul transmis ca parametru.
4. Creați următoarele obiecte (adăugați orice valoare pentru atributul *number*): 2 persoane cu numele "Mother" si "Father", 2 persoane cu numele "Jane", "John"; grupul "Parents", care conține persoanele "Mother" și "Father", grupul "Family", care conține grupul "Parents" și persoana "Jane". Folosind funcția **sendMessage**, să se trimită mesajul următor grupului "Family" și persoanei "John": "You are invited to my birthday party next week!". Memoria trebuie să fie gestionată corect.

SUBIECT Baze de date

Problema 1. (4 puncte)

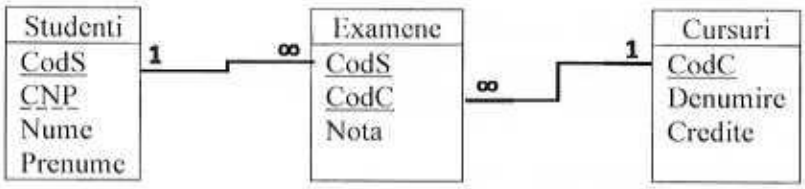
Un magazin care închiriază biciclete stochează următoarele informații într-o bază de date relațională:

- O firmă de biciclete are un cod, un nume și o țară de proveniență.
- O bicicletă aparține unei firme, are un cod, un model, o mărime (un număr, de ex.56) și un preț pentru închiriere pe zi.
- Un client are un CNP, un nume, un număr de telefon și o adresă de email.
- Când un client închiriază biciclete se crează o fișă de închiriere. Un client poate închiria mai multe biciclete deodată. O fișă de închiriere ține de un client, are o dată la care a fost făcută fișa și o listă de biciclete închiriate, fiecare cu o dată de început și de sfârșit pentru perioada de închiriere. Dacă un client revine în aceeași zi pentru a închiria biciclete suplimentare, atunci acestea se adaugă pe fișa existentă din acea zi. Un client nu are mai multe fișe de închiriere în aceeași zi. Dacă un client vine în altă zi să închirieze biciclete atunci se va crea o fișă de închiriere nouă.

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:
Clienti[CodClient, NumeClient, Telefon]
Soferi[CodSofer, NumeSofer, Salariu]
Autoturisme[CodAuto, NumarInmatriculare, MarcaAuto, *CodSofer*]
Curse[CodCursa, *CodClient, CodAuto*, DataCursa, MomentPlecare, DurataCursa, LocPlecare, Destinatie, SumaDePlata]
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, numele și salariul șoferilor care au cel mai mare salariu dintre șoferii care au avut cel puțin o cursă cu destinația „Iulius Mall Cluj" cu un autoturism marca „Ford". Eliminați duplicatele.

b. Se dau instanțele următoare ale relațiilor Autoturisme și Curse:

Autoturisme:

CodAuto	NumarInmatriculare	MarcaAuto	CodSofer
1	CJ11AAA	Ford	1
2	CJ22BBB	Opel	2
3	CJ33CCC	Volkswagen	3

Curse:

Cod Cursa	Cod Client	Cod Auto	Data Cursa	MomentPlecare	DurataCursa	Loc Plecare	Destinatie	Suma DePlata
1	1	1	2023.06.25	07:30	15	LP1	D1	20
2	1	2	2023.06.26	08:30	20	LP2	D3	30
3	2	1	2023.06.26	12:30	10	LP4	D4	15
4	1	1	2023.06.30	10:15	5	LP1	D2	10
5	3	3	2023.07.01	15:05	40	LP1	D4	50
6	3	2	2023.07.01	20:00	12	LP2	D2	15

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.CodSofer, COUNT(DISTINCT c.CodClient) Nr
FROM  Autoturisme a INNER JOIN Curse c ON a.CodAuto = c.CodAuto
WHERE c.DataCursa BETWEEN '2023.01.01' AND '2023.06.30'
GROUP BY a.CodSofer
HAVING COUNT(DISTINCT c.CodClient) >= ALL
        (SELECT COUNT(DISTINCT c.CodClient)
         FROM Autoturisme a INNER JOIN Curse c ON a.CodAuto = c.CodAuto
         WHERE c.DataCursa BETWEEN '2023.01.01' AND '2023.06.30'
         GROUP BY a.CodSofer)
```

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

- {CodCursa} → {CodAuto}
- {CodAuto} → {LocPlecare}.

SUBIECT Sisteme de Operare

Problema 1 (4p)

Răspundeți la următoarele întrebări despre execuția fragmentului de cod de mai jos, știind că funcția `mkfifo` creează un fișier FIFO, iar funcția `unlink` îl șterge.

```
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) Ce se va tipări în consolă?

b) Ce se va tipări în consolă dacă linia 18 se mută între liniile 9-10?

c) Cum va fi afectată funcționarea programului dacă linia 4 se mută între liniile 5-6?

d) În ce situație se va afișa 'a'?

e) Explicați rolul argumentelor funcției `mkfifo` de pe linia 4.

Problema 2 (5p)

Răspundeți la următoarele întrebări despre execuția scriptului Shell UNIX de mai jos.

```
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)	Care va fi conținutul fișierului `x/a.txt` din coloana din dreapta, după rularea scriptului cu argumentul `abc`?	`abc` `def`

b)	Care va fi conținutul fișierului `x/b.txt` din coloana din dreapta, după rularea scriptului cu argumentul `abc`?	`abc`
		`def`
		`abc`

c)	Care va fi conținutul fișierului `x/c.txt` din coloana din dreapta, după rularea scriptului cu argumentele `abc def`?	`def_def`
		`abc`
		`abc_def`
		`abc_abc`
		`def_def_def`

d)	Explicați în detaliu condiția de pe linia 7.	
e)	Explicați în detaliu expresia regulară de pe linia 8.	

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p

Problema 1. Punctaj - 1.5p

- Subalgoritm corect: **1p**
  - signatura corectă: 0.1p
  - implementare căutare binară: 0.9p
- Justificarea complexității (căutare binară): **0.5p**

Problema 2. Punctaj - 1.5p

- Subalgoritm corect implementat: **1.25p**
  - signatura corectă: 0.1p
  - implementare: 1.15p
- Complexitate $\theta(n)$: **0.25p**

Problema 3. Punctaj - 1.5p

Soluție: *șirul trebuie transformat în permutarea identică*.

- Soluție având complexitatea timp în caz defavorabil $\theta(n)$ - 1.5p
- Soluție având complexitatea timp în caz defavorabil $\theta(n \cdot \log_2 n)$ - 1p
- Soluție având complexitatea timp în caz defavorabil $\theta(n^2)$ - 0.75p
- Alte soluții care nu se încadrează în clasele de complexități de mai sus - 0.25p

Problema 4. Punctaj - 2.25p

- Implementarea clasei **SorterByTotalPrice** cu metoda statică **sort()** și folosirea funcției **std::sort**, cu o funcție lambda (**0.75p)**
  - pentru folosire **std::sort**, cu operator <
    bool operator<(Product*, Product*) 0.6p
  - pentru folosire **std::sort**, cu o functie standalone
    bool nume_functie(Product*, Product*) 0.75p
  - pentru funcții de sortare implementate (fără **std::sort**) corect 0.25p
- Implementarea metodei total_price in clasa **Computer** (**0.5p**)
- Implementarea constructorului în clasa **Computer** (**0.5p**)
- Implementarea metodei **toString()** in clasa **Computer** (**0.5p**)

Problema 5. Punctaj - 2.25p

- clasa ***Contact*** **(0.4p)**:
  - câmp privat *name*, constructor (0.1), metoda **getName()** (0.05), metoda abstractă **sendMessage** (0.15), destructor virtual (0.1).
- clasa **Person (0.35p):**
  - moștenire (0.05), câmp privat **number**, constructor (0.15), metoda **sendMessage** (0.15)
- clasa **Group (0.85p)**:
  - moștenire (0.05), câmp privat ***lista de contacte*** (0.2), constructor (0.1), metoda **addContact** (0.2), metoda **sendMessage** (0.3), destructor (a se vedea punctajul de mai jos pentru distrugerea corecta a obiectelor)
- main **(0.65p)**:
  - construire corectă obiecte **Person** (0.15), construire corectă obiecte **Group** + adăugare de contacte (0.3), apel **sendMessage** la Group si la Person (0.1), distrugere corecta obiecte (0.1)

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:

CodSofer	Nr
1	2

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

- **b2** - {CodCursa}→{CodAuto} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodAuto}→{LocPlecare} 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.

Subiect Sisteme de operare

Oficiu – 1p

Problema 1. Punctaj – 4p

a) Se va tipări 'x' – 0.5p
b) Se va tipări 'x' – 1p
c) Dacă părintele execută open înainte ca fiul să creeze fifo, fiul se blochează la open şi părintele la wait – 1p
d) Niciodată – 1p
e) Primul este numele fifo-ului de creat şi al doilea sunt permisiunile de acces – 0.5p

Problema 2. Punctaj – 5p

a) Conţinutul va fi acelaşi – 1p
b) Conţinutul va fi `def` urmat de `abc` pe altă linie – 1p
c) Conţinutul va fi `abc_def` urmat de `abc` pe altă linie urmat de `def` pe altă linie – 1p
d) Fişierul `$F` conţine cel puţin 2 linii formate exclusiv din una sau mai multe valori `$A` separate prin cel puţin câte un _ - 1p
e) Întreaga linie e o secvenţă care începe cu valoarea `$A` urmată de 0 sau mai multe secvenţe formate din unul sau mai multe _ urmate de valoarea `$A` – 1p