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

Proba scrisă a examenului de licență, 1 iulie 2019
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 Sisteme de operare

1 Răspundeți la următoarele întrebări, considerând că toate instrucțiunile din funcțiile C de mai jos se execută cu succes.

```
1  void f1(){
2    int i;
3    for(i=0; i<3; i++) {
4      if(fork() == 0) {}
5      wait(0);
6    }
7  }
8  void f2(){
9    int i, p = 0;
10   for(i=0; i<3; i++) {
11     if(p == 0) {
12       p = fork();
13     }
14     wait(0);
15   }
16 }
```

a) Desenați ierarhia proceselor create de execuția funcției `f1` în procesul părinte.

b) Desenați ierarhia proceselor create de execuția funcției `f2` în procesul părinte.

c) Rescrieți funcția `f1` astfel încât să creeze exact atâtea procese câte creează funcția `f2`, dar având altă ierarhie. Explicați și desenați ierarhia proceselor.

d) Explicați rolul apelului sistem `wait`.

2 Răspundeți la următoarele întrebări despre scriptul Shell UNIX de mai jos, considerând că este rulat într-un director cu structura dată alăturat, cu următoarele argumente în linia de comandă: `a d1 mere b d2 pere`.

```
1  #!/bin/bash
2  while [ -n "$1" ]; do
3    if [ -d $1 ]; then
4      for F in `find $1 -type f`; do
5        if file $F | grep -q -v text; then
6          echo "Eroare: $F"
7          continue
8        fi
9        sed "s/^$2//" $F > $F.text
10     done
11     shift
12     shift
13   else
14     shift
15   fi
16 done
```

```
.
├─ d1        - director
│  ├─ a      - fișier text
│  ├─ b      - fișier text
│  └─ x.txt  - fișier PDF redenumit .txt
└─ d2        - director
   └─ b      - fișier text
```

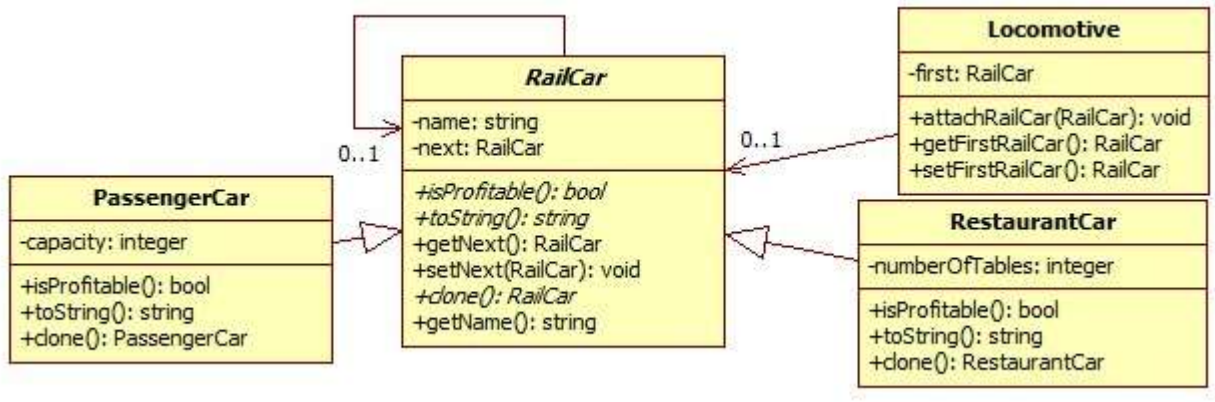
a) Ce afișează execuția scriptului și ce efect are asupra structurii de directoare și fișiere dată mai sus?
b) Explicați în detaliu linia 5
c) Cum se schimbă rezultatele execuției dacă pe linia 9 ghilimelele se înlocuiesc cu apostrofi?
d) Explicați ce se întâmplă dacă se șterg liniile 13 și 14.

VARIANTA 2

SUBIECT Algoritmică și programare

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

a) Definește clasele **RailCar** (Vagon), **PassengerCar** (Vagon persoane), **RestaurantCar** (Vagon restaurant), **Locomotive** (Locomotivă, memorează vagoanele atașate acesteia), pe baza următoarei diagrame UML (constructorii nu sunt indicați pe diagramă). Dintre metodele indicate pe diagramă, se vor implementa doar metodele indicate la punctul b), restul doar se declară.



- Atributul *name* din clasa **RailCar** (numele vagonului) trebuie să aibă cel puțin 2 caractere; atributele *capacity* (capacitate) din clasa **PassengerCar** și *numberOfTables* (număr mese) din clasa **RestaurantCar** trebuie să fie valori strict pozitive. Constructorii trebuie să impună constrângerile.
- Clasa abstractă **RailCar** are trei metode abstracte: **isProfitable, toString** și **clone**. Metodele **clone** din **PassengerCar** și **RestaurantCar** creează o copie a obiectului curent.
- Metoda **attachRailCar** din clasa **Locomotive** atașează vagonul parametru în succesiunea de vagoane deja atașate locomotivei astfel încât vagoanele să fi dispuse în ordine alfabetică după șirul de caractere returnat de metoda **toString.**
- Un vagon de persoane este profitabil (**isProfitable**) dacă are o capacitate (*capacity*) de peste 40 locuri. Un vagon restaurant este profitabil dacă numărul de mese din vagon (*numberOfTables*) este cel puțin 20.
- Pentru vagoanele de persoane metoda **toString** returnează numele (*name*) concatenat cu capacitatea (*capacity)*. Pentru vagoanele restaurant metoda **toString** returnează numele (*name*) concatenat cu numărul de mese (*numberOfTables*).

b) Implementează următoarele metode din diagrama de la punctul a): constructorii claselor **RailCar, PassengerCar**, **RestaurantCar**; metodele **toString** din clasele **PassengerCar** și **RestaurantCar**; metoda **attachRailCar** din clasa **Locomotive**; metodele **isProfitable** din clasele **PassengerCar** și **RestaurantCar**; metoda **clone** din **PassengerCar** și metoda **setNext** din **RailCar.**

c) Definește o funcție care primește ca parametru un obiect *l* de tip **Locomotive** și elimină din *l* acele vagoane care nu sunt profitabile.

d) Definește o funcție care primește ca parametru un obiect *l* de tip **Locomotive** și returnează un alt obiect de tip **Locomotive** conținând copii ale vagoanelor din *l* în ordine inversă.

e) Construiește în funcția principală (main) un obiect *l* de tip **Locomotive** în care se adaugă următoarele vagoane (alegeți valori pentru proprietățile lor neprecizate): trei obiecte de tip **PassengerCar** având 50, 32 și 40 de locuri și două obiecte de tip **RestaurantCar** având 25 și 20 mese. Eliminați din *l* vagoanele care nu sunt profitabile, folosind funcția de la c) și apoi apelați funcția de la punctul d) pentru a obține un nou obiect *l'* de tip **Locomotive**. La final, afișați vagoanele din *l'*.

f) Scrieți specificația metodei **attachRailCar** din clasa **Locomotive**.

- **Se va indica limbajul de programare folosit.**
- **Nu se vor defini alte metode decât cele specificate în diagramă (exceptând constructorii).**
- **Nu se vor folosi containere sortate și operații de sortare predefinite.**

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

VARIANTA 2

SUBIECT Baze de date

Fie o bază de date care stochează informații despre istoricul pacienților dintr-un spital. Se știe faptul că în spital nu sunt mai mulți doctori cu același nume și prenume. Baza de date are următoarea structură:

- tabelul ***Pacienti*** cu câmpurile **CodP, Nume, Prenume, Varsta, DataNasterii**;
- tabelul ***Proceduri*** cu câmpurile **CodProc**, **Nume, Descriere, Pret**;
- tabelul ***FisePacienti*** cu câmpurile **CodPacient**, **CodProcedura**, **Data, NumeDoctor**, **PrenumeDoctor, SpecializareDoctor, VechimeDoctor**.

1. Determinați cheile primare și cheile externe pentru fiecare dintre tabelele de mai sus.
2. Determinați cel puțin 3 dependențe funcționale care se referă la câmpuri care nu reprezintă coduri.
3. Considerați următoarele modificări de structură și precizați care dintre ele sunt esențiale pentru ca baza de date să fie în **3NF** (a 3-a formă normală). În cazul unui răspuns afirmativ, justificați-vă opțiunea.
    a. Crearea unui tabel separat pentru stocarea doctorilor.
    b. Adăugarea constrângerii de integritate **Varsta = data curenta – DataNasterii** în tabelul ***Pacienti.***
    c. Eliminarea câmpului **Varsta** din tabelul ***Pacienti***.
    d. Crearea unui tabel separat pentru stocarea vârstei pacienților.
    e. Adăugarea constrângerii de integritate **Data** < **data curenta** în tabelul ***FisePacienti.***
4. Scrieți, pe structura dată, o interogare SQL echivalentă cu următoarea interogare:

$\Pi_{\text{Nume, Prenume}} (\sigma_{\text{SpecializareDoctor = 'radiologie'}} (\textbf{\textit{FisePacienti}}) \otimes_{\text{CodPacient = CodP}} \textbf{\textit{Pacienti}})$
$\Pi_{\text{Nume, Prenume}} (\sigma_{\text{SpecializareDoctor = 'cardiologie'}} (\textbf{\textit{FisePacienti}}) \otimes_{\text{CodPacient = CodP}} \textbf{\textit{Pacienti}})$

5. Scrieți, pe structura dată, o interogare SQL care returnează, pentru specializările la care s-au încasat cei mai mulți bani din proceduri, numele specializării, numărul total de proceduri executate în cadrul specializării și suma încasată (***Specializare, NrProceduri, Sumă***). Se vor afișa toate specializările care îndeplinesc această condiție.

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p
Definirea clasei abstracte **RailCar** – 1 p din care
atribut – 0.2
constructor si metode **-** 0.8
Definirea clasei **PassengerCar**– 1.1 p din care
relația de moștenire – 0.2
atribut – 0.1
constructor si metode – 0.8
Definirea clasei **RestaurantCar** – 1.1 p din care
relația de moștenire – 0.2
atribut – 0.1
constructor si metode – 0.8
Definirea clasei **Locomotive** – 1.3p din care
atribute– 0.1
constructor - 0.1
metoda attachRailCar - 1.1p
Funcția de la punctul c) – 1.8p din care
signatura corectă - 0.1p
eliminare vagoane neprofitabile- 1.7p
Funcția de la punctul d) – 1.7p din care
signatura corectă - 0.1p
creare lista vagoane - 1.6p
Funcția principală e) – 0.5p
f) Specificaţia metodei **attachRailCar** din clasa **Locomotive**– 0.5p

Notă. Datorită faptului că signatura metodei setFirstRailCar() este incompletă pe diagrama UML, se va acorda punctaj maxim (1.7p) pentru punctul d).

Subiect Baze de date
1. 0.5p (chei primare) + 0.5p (chei externe) = 1p
2. 1p
3. a, c
2 x (0.5p răspuns + 0.5p justificare) = 2p
4. rezolvarea completă a interogării = 2p
5. rezolvarea completă a interogării = 3p
1p of

Subiect Sisteme de operare

Oficiu – 1p

1.a Diagramă cu proces părinte, fii, nepoți și strănepoți, totalizând 8 procese – 1p
1.b Diagramă liniară cu 4 procese: părinte, fiu, nepot, strănepot – 1.5p
1.c Implementare corectă - 0.5p
Explicație – 0.5p
Diagramă – 0.5p
1.d Așteaptă terminarea unui proces fiu – 1p

2.a Afișează eroare pentru x.txt și creează trei fișiere cu extensia .text din care se șterg cuvintele date – 1.5p
2.b Comenzile file și grep – 0.5p
Argumentele –q și –v – 0.5p
Expresia regulară – 0.5p
2.c Caută literalmente $2, fără a substitui valoarea argumentului – 0.5p
2.d Ciclu infinit – 0.5p