Logică şi structuri discrete

1. Presupunem că există 200 de studenţi din anul I care participă la cursurile de calculatoare, matematică şi fizică. Rezultatele au arătat că 90 de studenţi participă la calculatoare, 110 la matematică şi 60 la fizică. Mai departe 20 de studenţi participă la calculatoare şi matematică. 20 de studenţi participă la calculatoare şi fizică şi 30 la matematică şi fizică. Ne interesează să aflăm cati studenţi participă la toate cele trei cursuri.

2. Folosind regulile de inferenţă din logica propoziţională, să se stabilească valoarea de adevăr a următoarei afirmaţii: *Dacă fie soţia mea fie eu am fi adus banii cu noi, atunci am fi putut plăti şi cina la restaurant şi taxiul. Dacă am fi plătit taxiul n-ar fi trebuit să mergem pe jos acasă. Dar a trebuit să mergem pe jos acasă. Deci, soţia mea nu a adus banii.*

3. Să se aplice rezoluţia pentru a rezolva următoarea problemă de inferenţă:
Să presupunem că:
(a) Există un dragon.
(b) Dragonul doarme în peştera sa sau vânează în pădure.
(c) Dacă dragonul este flămând, atunci nu poate dormi.
(d) Dacă dragonul este obosit, atunci nu poate vâna.

(i) Ce face dragonul când este flămând?

Tehnici de programare

1. Daţi o soluţie recursivă, în limbajul de programare C, pentru Algoritmul lui Euclid de aflare a celui mai mare divizor comun a două numere întregi, pozitive.
Formularea în cuvinte a algoritmului este următoarea:
Dacă unul dintre numere este zero, c.m.m.d.c. al lor este celălalt număr.
Dacă nici unul dintre numere nu este zero, atunci c.m.m.d.c. nu se modifică dacă se înlocuieşte unul dintre numere cu restul împărţirii sale cu celălalt.

2. Să se scrie o funcţie care inversează in situ conţinutul unui şir de caractere. Se vor folosi operaţii cu pointeri.

3. Folosind operaţiile pe biţi din limbajul de programare C, să se scrie o funcţie, `contorbiti`, care să numere biţii cu valoarea `1` din argumentul său `x`, de tipul întreg fără semn.

Programarea orientată pe obiecte

1. Să se creeze o clasă MyException derivată din clasa Exception ce conţine:

- un constructor ce are ca parametru un şir de caractere ce va fi furnizat de serviciul getMessage() al excepţiei. Serviciul getMessage() nu va fi suprascris.

- o metodă ce returnează de câte ori a fost instanţiată, atât ea (clasa MyException) cât şi orice subclasă a sa. După ce aţi implementat metoda precum şi mecanismul de numărare, explicaţi datorită cărui fapt metoda returnează şi câte instanţe ale subclaselor au fost create.

Creaţi într-o metodă main trei obiecte de tip MyException care vor fi ataşate pe rând aceleiaşi referinţe. Apelaţi pentru fiecare obiect creat două servicii furnizate de acesta.

2. Folosind clasa ArrayList creaţi o clasă Biblioteca ce poate stoca un număr nelimitat de obiecte de tip Carte. O carte are două atribute ce stochează titlul precum şi autorul cărţii iar afişarea acesteia pe ecran va furniza utilizatorului valorile atributelor menţionate.
Clasa Biblioteca oferă doar două servicii, unul pentru adăugarea de elemente de tip Carte şi altul pentru afişarea elementelor conţinute. Se cere implementarea claselor menţionate precum şi crearea într-o metodă main a unei biblioteci ce are trei cărţi. Cărţile ce există în bibliotecă vor fi tipărite.

3. Respectând cerinţele enunţate şi principiile POO, să se implementeze în Java interfaţa
şi clasele descrise mai jos.
Interfaţa Intrare conţine:
- o metodă denumită continut, fără argumente şi care returnează o referinţă String
Clasa Fisier implementează interfaţa Intrare şi conţine:
- un atribut de tip String denumit informatie, specific fiecărui obiect Fisier în parte
- un constructor ce permite setarea atributului anterior cu o valoare String dată ca parametru constructorului
- implementarea metodei continut întoarce valoarea atributului informatie descris mai sus
Clasa Director implementează interfaţa Intrare şi conţine:
- un atribut denumit intrari de tip ArrayList<Intrare>; câmpul este specific fiecărei instanţe a acestei clase si se va iniţializa cu un obiect listă gol
- o metoda adauga cu un singur parametru; acesta trebuie să fie declarat în aşa fel încât să poată referi atât obiecte a clasei Director, cât şi obiecte a clasei Fisier dar să NU poată referi orice fel de obiect Java (ex. NU va putea referi un obiect String); metoda introduce în lista anterioară referinţa primită ca parametru
- obligatoriu în implementarea metodei continut se parcurge lista intrari şi se apelează metoda continut pe fiecare referinţă din lista concatenându-se String-urile întoarse de aceste apeluri; metoda va returna o referinţă spre String-ul rezultat în urma concatenării
În toată această ultimă clasă, obiectele Fisier si obiectele Director din listă trebuie să fie tratate uniform.

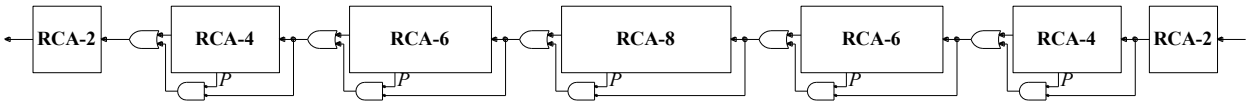
Arhitectura calculatoarelor

1. Se consideră următorul format ipotetic inspirat de formatul de virgulă flotantă IEEE 754, prezentat prin lungimea câmpurilor specifice în următorul tabel:

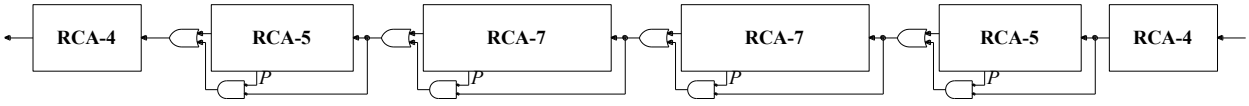
Semn	Exponent	Mantisă
<1 bit>	<5 biţi>	<10 biţi>

Considerând ca normalizarea se face similar formatului IEEE 754, se cere să se calculeze cel mai mic si cel mai mare număr pozitiv normalizat care se pot reprezenta utilizând formatul propus.

2. Se consideră un sumator Carry Skip (CSkA) pe 32 de biţi, realizat cu segmente Ripple Carry Adder (RCA) inegale, în două variante, ca în figura de mai jos:



a)



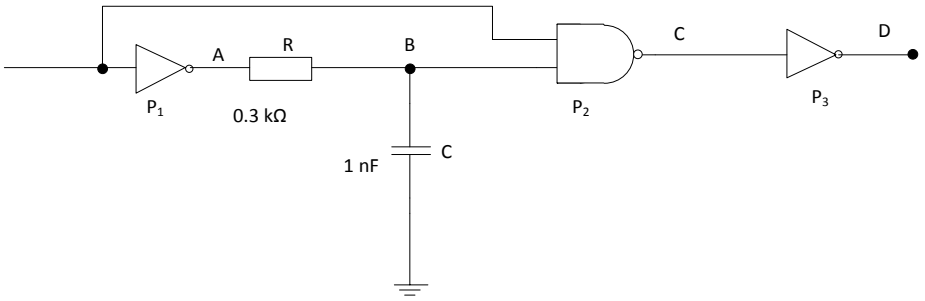
b)

Să se evalueze latența maximă de generare a semnalului *carry out* și a *sumei* pentru cele 2 variante în termenii latenței pe o poartă logică (latența pentru o poartă EXOR este dublă față de cea pentru o poartă convențională).

3. Să se înmulteasca după procedura lui Booth numerele -101 si -11, organizând rezolvarea sub formă tabelară, prin indicarea pas cu pas a conţinutului regiştrilor şi a semnalelor de control activate.

Circuite şi semnale numerice

1. Se da schema din fig. 1
   a. Se cere sa se ridice diagramele de timp din punctele V_i, V_A, V_B, V_C, V_D.
   b. Sa se determine duratele de timp de la iesirea V_D pentru urmatoarele: P_1, P_2, P_3 sunt porti TTL, V_H = 3,5 V, V_L = 0,2 V, V_T = 1,5 V, T_1=T_2=10µs. Timpul de propagare pe porti se poate neglija in raport cu ceilalti timpi din figura.



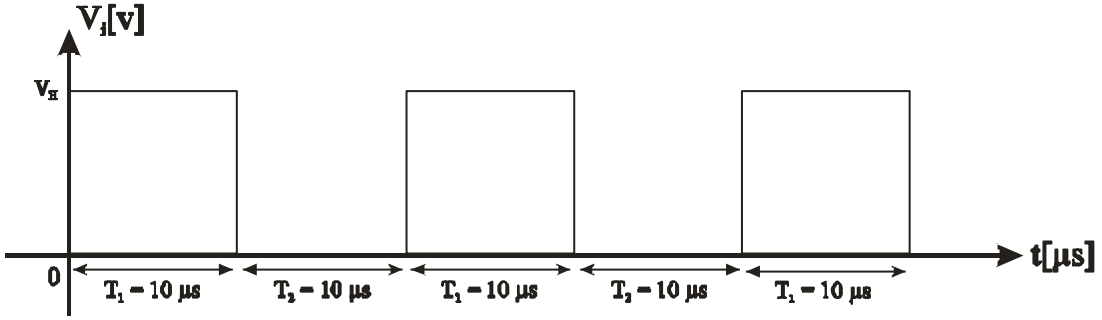
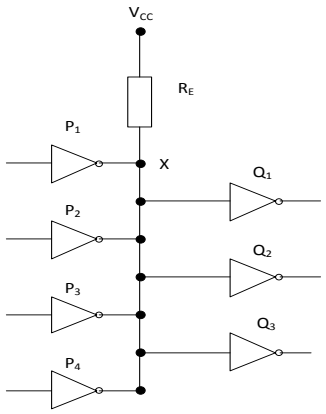


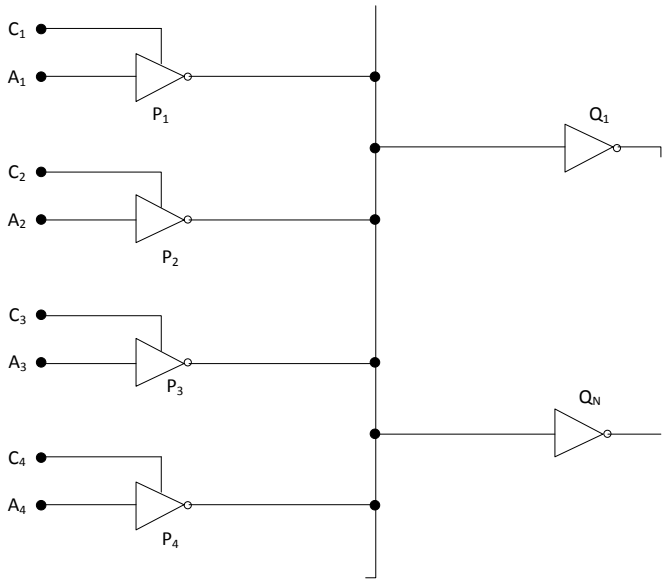
Fig. 1

2. Se da urmatoarea schema:



Se cere sa se dimensioneze rezistenta R_E, unde portile P_1, P_2, P_3, P_4 sunt porti TTL cu collector in gol, iar Q1, Q_2, Q_3 sunt porti TTL standard.
Se da: V_{CC} = 5 V, V_{OH} = 3,5 V, V_{OL} = 0,2 V, I_{OL} = 16 mA, I_{00} = +250 μA, I_{iL} = -1,6 mA, I_{iH} = 40 μA, I_{C0} = 0.

3. Se da următoarea schema de interconectare la o magistrala



Portile P_1, P_2, P_3, P_4 sunt porti cu impedanta ridicata din cadrul familiei de circuite TTL. Pentru a nu exista coliziuni pe magistrala o singura poarta P trebuie sa furnizeze valoarea logica “0” sau “1”, celelalte porti P trebuie sa fie pe impedanta ridicata. Fie parta P_1 activa si P_2, P_3, P_4 sunt comandata cu semnalele C_2, C_3, C_4 sa functioneze cu iesirile pe impedanta ridicata.
Se cere sa se specific cate porti TTL standard (Q) mai pot fi conectate pe magistrala de mai sus.

Se da pentru portile TTL cu impedanta ridicata : I_{OH} = -5,2 mA, I_{OL}= 16 mA, I_{OLZ} = -40 μA, I_{OHZ} = 40 μA. Pentru portile TTL din deria standard avem: I_{IL} =-1,6 mA, I_{IH} =40 μA.
Se aminteste ca litera O se refera la iesire, I se refera la intrare, Z la impedanta ridicata, semnul negativ semnifica sensul se iesire din poarta (la intrare/iesire)

Proiectarea şi analiza algoritmilor

1. Să se scrie funcţia de suprimare a unui nod dintr-un arbore binar ordonat implementat cu pointeri. Se vor trata cele trei cazuri specifice: i) nodul de şters nu are fii, este terminal; ii) nodul de şters are un singur fiu; iii) nodul de şters are ambii fii. Pentru cazul iii) se va alege nodul înlocuitor ca fiind maximul din subarborele stâng sau minimul din subarborele drept al nodului de şters.

2. Să se scrie structura de date şi funcţia ce afişează factorul de umplere pentru fiecare pagină a unui arbore B de ordin N. Factorul de umplere este dat de raportul dintre numărul de chei dintr-o pagină şi capacitatea maximă a paginii. Pentru o pagină a unui arbore de ordinul 2, ce conţine 3 chei procentul de umplere este 0.75.

3. Să se scrie rutinele pentru: i) inserţia unui nod de tip PARCARE într-o listă înlănţuită de parcări modelată prin nume; ii) inserţia unui nod de tip AUTOMOBIL modelat prin număr de înmatriculare de forma AA-00-AAA (de exemplu TM-01-UPT) la un nod PARCARE creat anterior şi identificat prin nume.

Fundamente de inginerie software

1. Pentru codul de mai jos desenati diagrama de secventa incepand cu apelul din linia marcata cu *** si pana la revenirea din acel apel. **Nota**: Apelurile catre System.out se ignora. In plus, NU trebuie trasate bare de activare si valori returnate

```
interface DispozitivAlarma {
      public int start();
}
class Senzor {
      private DispozitivAlarma alarma;
      public Senzor(DispozitivAlarma alarma) { this.alarma = alarma; }
      private void declanseaza() {
            int i = alarma.start();
            System.out.println("Informari:" + i);
      }
      public void alarmare() { declanseaza(); }
}
class Simplu implements DispozitivAlarma {
      public int start() {
            System.out.println("Alarma!");
            return 1;
      }
}
class Lant implements DispozitivAlarma {
      private DispozitivAlarma next;
```

```
      public Lant(DispozitivAlarma disp) { this.next = disp; }
      public int start() {
            proceseaza();
            return (1 + next.start());
      }
      public void proceseaza() {
            System.out.println("Alarma intermediara!");
      }
}
class Test {
      public static void main(String argv[]) {
            Simplu s  = new Simplu();
            Lant   l1 = new Lant(s);
            Lant   l2 = new Lant(l1);
            Lant   l3 = new Lant(l2);
            Senzor sen= new Senzor(l3);
            /***/sen.alarmare();
      }
}
```

2. Sa se scrie o suita de teste white-box (glass-box) prin care sa se faca testarea cailor pentru pseudo-codul de mai jos. Justificati alegerea facuta.

Nota: se puncteaza doar testcase-urile concrete justificate. Discutia abstracta, fara date de test concrete nu se puncteaza.

Nota: nu se cere deloc implementare in jUnit / cppUnit!

```
int error;
int max(int tab[], int entries) {
     if(entries <= 0) {
          error = 1;
          return -1;
     }
     error = 0;
     int max = tab[0];
     for(int i = 1; i < entries; i++) {
          if(max < tab[i]) {
          max = tab[i];
          }
     }
     return max;
}
```

3. Se da urmatorul document de specificare a cerintelor pentru un sistem de gestionare a unui magazin:

Cumparatorii vor avea acces on-line la sistem, pentru a consulta oferta de produse a magazinului. La cerere, un cumparator poate alege online un numar de produse si poate cere tiparirea descrierii acelor produse, incluzand si pretul fiecarui produs. Vanzatorii vor accesa sistemul pentru a realiza, daca este posibil, marcarea ca rezervata a unui anumit produs, pe o anumita cantitate specificata. In cazul efectuarii rezervarii, sistemul va calcula costul, pe baza unor tabele de tarife. Vanzatorul va confirma plata in avans, iar sistemul va tipari un bon. Cumparatorii, prin intermediul vanzatorilor, vor avea posibilitatea sa renunte total sau partial la o rezervare de produse, caz in care sistemul va trebui sa calculeze suma de rambursat.

Gestionarea bazei de date cu informatii legate de produse si preturi va fi efectuata de catre un sef de raion. Informatiile despre produse vor putea fi modificate si de catre vanzatori. Toate operaţiile cu excepţia consultării ofertei de produse de către cumparator presupun o loginare prealabilă in sistem.

Se cere:

a. Identificaţi toţi actorii şi use-case-urile (user stories) pentru sistemul descris mai sus. Justificaţi alegerea făcută!
b. Reprezentaţi actorii şi use-case-urile identificate în UML sub forma unei Use-Case Diagram
c. Daca dorim sa "dam factor comun" descrierea functionalitatii de loginare intre diferitele use-case-uri care o contin, cum se reprezenta acest lucru intr-o Use-Case Diagram?

Fundamente de ingineria calculatoarelor

1. Se da urmatorul cod MIPS care operează pe întregi:

```
L: ADD      R1, R2, R3
SUB         R3, R1, R2
AND         R2, R2, R5
SW          0(R2), R3
SW          16(R2), R1
XOR         R9, R7, R2
BZ          R1, L
```

Aratati, marcand fiecare artificiu utilizat, precum si fiecare stall, cum se executa codul in pipeline-ul MIPS daca:

- Beneficiem doar de artificiul cu tactul
- Stagiul de MEM pentru instructiunile de tip Load/store se executa intr-o unitate distincta (e diferit de MEM-ul corespunzator celorlate instructiuni) si dureaza 2 cc in loc de 1, neparalelizabil (ca si la DIV). Notati aceste doua impulsuri cu MEM1, respectiv MEM2
- Branch-ul este tratat in maniera freeze & flush si se decide in ciclul de EX

2. Se dă următorul cod. Se consideră pipeline-ul MIPS standard, beneficiind de artificiul cu tactul, forwarding şi bypassing. Unităţile de multiply au nevoie de 4 cc pt execuţia efectivă a înmulţirii, în loc de 7(standard), iar cele de adunare de fp au nevoie doar de 2cc, în loc de 4 (standard). Branch-urile se tratează în maniera predict as taken.

```
      XOR R3,R3,R3
foo   L.D F6, 0(R1)
      L.D F8, 0(R2)
      MUL.D F10, F6, F8
      ADD.D F4, F4, F10
      ADDI R1, R1, #-8
      ADDI R2, R2, #-8
      ADDI R3, R3, #16
      BZ R3, foo
```

Arătaţi cum se execută următorul cod în pipeline. Marcati toate artificiile folosite si stall-urile, iar la branch precizați care este ramura folosită.

3. Se dă următorul cod. Se consideră pipeline-ul MIPS standard, beneficiind de artificiul cu tactul, forwarding şi bypassing. Unităţile de multiply au nevoie de 4 cc pt execuţia efectivă a înmulţirii, în loc de 7 (standard), iar cele de adunare de fp au nevoie doar de 3cc, în loc de 4 (standard). Branch-urile se tratează în maniera predict as taken.

```
      SUB R3, R3, R3
foo   ADDI R1, R1, #-8
      L.D F8, 0(R2)
      L.D F6, 0(R1)
      ADD.D F10, F6, F8
      ADDI R2, R2, #-8
      MUL.D F4, F4, F10
      ADDI R3, R3, #-1
      BGEZ R3, foo
```

Arătaţi cum se execută următorul cod în pipeline. Marcati toate artificiile folosite si stall-urile, iar la branch precizaţi care este ramura folosită.

Sisteme de operare

1. Scrieti un program care rulează, una câte una, comenzile date ca şi argumente în linia de comandă până când una din ele întoarce o valoare de ieşire diferită de zero.
Programul va fi scris în limbajul C, folosind apelurile sistem şi funcţiile din biblioteca standard POSIX. Nu este necesar să includeţi fişiere antet.
Indicaţii:
pot fi utile:

```
int execl(char *file, const char *arg0, ... );

//idem execlp

pid_t fork(void);

pid_t wait (int *status);
```

2. Scrieţi un program care execută două comenzi primite ca şi argumente în linia de comandă astfel încât ieşirea standard a primei comenzi să ajungă să fie citită la intrarea standard a celei de-a doua.
De exemplu:

```
./program ls sort
```

va rula comenzile ls si sort similar cu linia de comanda shell:

```
ls | sort
```

Programul va fi scris în limbajul C, folosind apelurile sistem şi funcţiile din biblioteca standard POSIX. Nu este necesar să includeţi fişiere antet.
Indicaţii:
pot fi utile:

```
int execl(char *file, const char *arg0, ... );
//idem execlp
int dup2(int old, int new);
int pipe(int fildes[2]);
```

```
pid_t fork(void);
pid_t wait (int *status);
```

3. Să se scrie un program format din două procese, cu comportamentul descris în continuare. Aproximativ o dată la secundă, primul proces îl informează pe al doilea de scurgerea acestui interval de timp. Al doilea proces numără continuu începând de la 1, iar când se adună un număr de 7 notificări de la primul proces, afişează numărul curent şi se termină.
Programul va fi scris în limbajul C, folosind apelurile sistem şi funcţiile din biblioteca standard POSIX. Nu este necesar să includeţi fişiere antet.
Indicatie:
semnalele utilizator sunt SIGUSR1 si SIGUSR2;
pot fi utile:

```
typedef void (*sighandler_t)(int);

sighandler_t signal(int signum, sighandler_t handler);

pid_t fork(void);

pid_t wait(int *status);

int kill(pid_t pid, int sig);

unsigned int sleep(unsigned int seconds);
```

Bazele inteligenţei artificiale

1. Se cere
   - Descrieţi succint tehnica căutării în adâncime într-un graf definit printr-un obiect compus de forma:
     Arce is [m(a,b), m(b,d), m(a,c), m(b,e),...]
   - Reprezentaţi graphic procedura de căutare în adâncime într-un graf.
   - Scrieti utilizând limbajul Prolog procedura de căutare în adâncime într-un graf.

2. Se cere
   - Descrieţi succint tehnica căutării în adâncime într-un graf definit printr-un set de axiome de forma:
     s(a,b). s(b,d). s(a,c). s(b,e)....
     atunci când adâncimea de căutare este limitată
   - Reprezentaţi graphic procedura de căutare în adâncime într-un graf când adâncimea de căutare este limitată
   - Scrieti utilizând limbajul Prolog procedura de căutare în adâncime într-un graf când adâncimea de căutare este limitată

3. Se cere

- Descrieţi succint tehnica căutării în lătime într-un graf definit printr-un set de axiome de forma:

  s(a,b). s(b,d). s(a,c). s(b,e)....
- Reprezentaţi graphic procedura de căutare în lăţime într-un graf.
- Scrieti utilizând limbajul Prolog procedura de căutare în lăţime într-un graf.

Sisteme încorporate

1. Prezentaţi soluţia pentru conectarea unui circuit de memorie externă de 64 KO la un microcontroler care nu are magistrale externe.

2. Desenaţi schema pentru comanda a 8 LED-uri şi citirea a 8 semnale externe de către un microcontroler 8051.

3. Prezentaţi soluţia pentru conectarea unui circuit de memorie de 64 KO ca memorie externă de program şi a unui alt circuit de 64 KO ca memorie externă de date.

Baze de date

Nota: Pentru rezolvarea problemelor propuse se poate folosi şi limbajul SQL sau PL/SQL, dar cu definirea cheilor primare şi externe.

1. Se consideră două fişiere dintr-o bază de date existentă pentru rezervare locuri la avion. Fişierul **Curse** conţine datele pentru fiecare cursă din orarul de zbor.
Fişierul **Pasageri** conţine datele pentru toţi pasagerii din toate cursele. Toţi pasagerii unei curse vor avea acelaşi cod cursă CodC.

Curse

CodC	**Pilot**	**Copilot**	**Avion**	**Oras1**	**Ora1**	**Oras2**	**Ora2**	**PretR**	**NrLoc**

↓ **Pasageri**

CodC	**CNP**	**NumeP**	**Adresa**	**DataN**	**Tel**	**Pret**

Să se afişeze informaţiile despre o cursă de avion şi lista pasagerilor din acea cursă folosind comenzi simpe din limbajul XBase. Se recomandă forma de afişare:

CodC	Oras1	Ora1	Oras2	Ora2	Pilot	Copilot	TipAv
Ro234-0609	**Timisoara TSR**	**8:30**	**Bucuresti**	**9:30**	**Popescu**	**Adam**	**B747**

Lista Pasageri

CNP	**Nume Pas**	**Adresa**	**Telefon**	**DataN**	**Pret**

2. Considerăm o bază de date normalizată pentru evidenţă studenţi care cuprinde tabelele:

STUD

CodS	**Nume**	**Adresa**	**DataN**	**Bursa**	**Telefon**	**CNP**	**.....**

NOTE

CodS	**CodC**	**NOTA**	**Data**

CURS

CodC	**Titlu**	**NumeProf**

Folosind comenzi simple Xbase se cere afişarea notelor unui student dat prin nume sub forma:

CODS: AC321 Nume: Popescu Bogdan Adresa: Bogdanesi 3 Bursa:150

Situatia notelor

Curs	Nota	Data	Nume prof
Sisteme de operare	9	23-01-2011	Popovici

.....

3. Considerăm o bază de date normalizată pentru evidenţă studenţi care cuprinde tabelele**:**

STUD

CodS	**Nume**	**Adresa**	**DataN**	**Bursa**	**Telefon**	**CNP**	**.....**

NOTE

CodS	**CodC**	**NOTA**	**Data**

CURS

CodC	**Titlu**	**NumeProf**

Folosind comenzile XBase elementare pentru dialog şi SQL pentru căutarea informaţiilor, să se scrie secvenţe de program care realizează funcţiile:

- Afişare Cods, Nume student, Bursa, pentru toţi studenţii care au medii mai mari decât o valoare N
- Afişare pentru un student dat prin Cods toate notele, Titlul cursului pentru fiecare notă şi Nume profesor.

Se va ţine cont că rezultatul unei comenzi SQL se obţine într-o zona de lucru din care se poate afişa direct folosind comanda BROWSE, iar variabilele citite sunt externe precedate de "**:**".