BABEŞ-BOLYAI TUDOMÁNYEGYETEM

MATEMATIKA ÉS INFORMATIKA KAR

Záróvizsga – 2015 június
Magyar informatika szak

1. tétel

Írjunk egy programot Python, C++, Java, vagy C# nyelvekben, mely:

a) Egy *Termék* osztályt **definiál**, egy privát egész típusú *termékÁr* mezővel, egy publikus konstruktorral, mely a *termékÁr*at inicializálja, és egy publikus *eladóiÁr*() metódussal, mely a termék árát téríti vissza.

b) Egy *csomagoltTermék* osztályt **definiál**, melyet a *Termék* osztályból származtatunk és tartalmaz egy privát egész típusú *csomagolásiÁr* mezőt, egy publikus konstruktort, mely a *termékÁr,* és a *csomagolásiÁr* attribútumokat inicializálja, valamint egy felüldefiniált *eladóiÁr*() metódust, mely a *termékÁr* és a *csomagolásiÁr* mezők összegét téríti vissza.

c) Egy függvényt **definiál**, mely egy *Termék* típusú objektumot egy, az *eladóiÁr* szerint, csökkenően rendezett *Termék-listába* szúr be (a lista a beszúrás után is rendezett marad).

d) A c-pontban definiált függvény segítségével **definiáljunk egy függvényt**, mely egy háromelemű, *eladóiÁr* szerint rendezett *Termék*-listát térít vissza a következő tartalommal: egy *Termék*-et melynek *termékÁr*a 100, egy *csomagoltTermék*-et, melyek *termékÁra* 70, *csomagolásiÁra* 10, egy *csomagoltTermék*-et, melyek *termékÁra* 90, *csomagolásiÁra* 15.

e) Egy függvényt **definiál**, mely egy, a d-pontban definiált *Termék*-lista bemenő paraméterre visszatéríti a termékek *eladói árainak* átlagát.

f) Tartalmazza a **főprogramot**, mely a d-pontbeli függvényt hívja, majd a standard outputra (konzolra) kiírja az e-pontbeli függvény visszatérítési értékét.

g) Specifikáljuk a programban használt *lista* adattípussal kapcsolatos műveleteket.

Megjegyzések:

- Ne használjunk rendezett konténereket;
- Ne definiáljunk a tételben kért metódusok mellett más metódusokat;
- Ne használjunk listarendezést.

*Lehet használni létező **Lista**-könyvtárakat (Python, C++, Java, C#). Amennyiben nem egy standard könyvtárat használunk, minden függvény interfészét specifikáljuk.*

2. tétel

a. Tervezzünk **relációs adatbázis sémát,** melynek táblái **3NF**-ban vannak egy receptkönyv következő információira:

- **hozzávalók**: hozzávaló megnevezése, kategóriája (lisztféleségek, édesítők, aromák, stb.), mértékegysége (gramm, liter, stb.), kalóriája egy egységre vonatkozóan;
- **sütemények**: sütemény kódja, neve, leírása, hozzávalók listája (a megfelelő mennyiséggel minden hozzávalóból).

Indokoljuk, hogy a végső táblák 3NF-ban vannak.

b. A SELECT-SQL parancsot és legalább egyszer relációs algebrát használva, az **a.** pont adatbázisára vonatkozóan adjuk meg:
b1. A „Tiramisu” hozzávalóit (megnevezése, kategóriája, mértékegysége, mennyisége.
b2. Azokat a süteményeket (kód, név), melyek tartalmaznak édesítőt, és nem tartalmaznak vanília aromát.
b3. Azokat a süteményeket (kód, név), melyek kalóriaértéke kevesebb, mint 500.

3. tétel

3.1. Feltételezzük, hogy a következő programrészletben minden utasítás hiba nélkül fut le, és hogy a pipe-okat helyesen lezártuk. Válaszoljunk a következő kérdésekre!

```
1  #define N 6
2  #define K 1
3
4  int main() {
5    int i, n=1, p[N][2];
6    for(i=0; i<N; i++) { pipe(p[i]); }
7    write(p[0][1], &n, sizeof(int));
8
9    for(i=0; i<N; i++) {
10     if(fork() == 0) {
11       read(p[i][0], &n, sizeof(int));
12       n++;
13       write(p[(i+K)%N][1], &n, sizeof(int));
14       exit(0);
15     }
16   }
17
18   for(i=0; i<N; i++) { wait(0); }
19   read(p[0][0], &n, sizeof(int));
20   printf("%d\n", n);
21   return 0;
22 }
```

(a) Hány gyerekfolyamatot hozunk létre?

(b) Mi jelenik meg a képernyőn? Magyarázzuk meg a program-részlet működését!

(c) Magyarázzuk meg a program-részlet működését K=2 esetén!

3.2. Tekintsük az alábbi UNIX shell script-et.

(a) Magyarázzuk meg a script működését!

(b) Mi jelenik meg a képernyőn?

(c) Magyarázzuk el részletesen a 8. sort!

(d) Magyarázzuk meg miért van szükség a „\” karakterre a 11. sorban!

```
 1  #!/bin/bash
 2
 3  M=0
 4  N=0
 5  for F in *.log; do
 6      A=`grep "\<ERROR\>" $F | wc -l`
 7      M=`expr $M + $A`
 8      B=`grep "\<ERROR\>.*\<segmentation fault\> " $F | wc -l`
 9      N=`expr $N + $B`
10  done
11  expr 100 \* $N / $M
```

Megjegyzés: Az összes tétel kötelező.
Minden tételt 1-től 10-ig osztályoz két javító.
Munkaidő: 3 óra.

BAREM

INFORMATICĂ

Subiect 1 (Algoritmică şi Programare):
Oficiu – **1**p
Definirea clasei *Produs*– **0.75**p din care
- atribut – 0.25
- constructor – 0.25
- metoda *preţDeVânzare*() - 0.25

Definirea clasei *ProdusAmbalat*– **1.75**p din care
- relația de moștenire – 0.25
- constructor – 0.5
- atribut – 0.25
- metoda *preţDeVânzare*() – 0.75

Funcția de la punctul c) – **2.25**p din care
- signatura corectă - 0.25p
- algoritmul de inserare în interiorul listei - 1.75p
  - parcurgere listă și determinarea poziției de inserare – 1.25p
  - adăugare element pe poziția determinată anterior – 0.5
- returnare rezultat - 0.25p

Funcția de la punctul d) – **1**p din care
- signatura corectă și declarare listă- 0.25p
- creare obiecte – 0.25p
- inserare obiecte în listă - 0.25p
- returnare rezultat - 0.25p

Funcția de la punctul e) – **1.25**p din care
- signatura corectă - 0.25p
- parcurgere listă cu calcul preț total de vânzare – 0.75 p
- returnare rezultat - 0.25p

Program – **0.5**p din care
- apel funcție construire listă – 0.25p
- afișarea rezultatului cerut – 0.25p

Specificațiile operațiilor folosite din tipul de dată *Listă*– **1.5**p

Subiect 2 (Baze de date):
1 punct oficiu
Problema a:
- **1 punct** pentru dependențe funcționale
- **1 punct** pentru tabelele în 3NF;
- **1 punct** pentru justificare.

Problema b:
- **2 puncte** pentru b1
- **2 puncte** pentru b2
  - **0.5 puncte** pentru prajiturile cu îndulcitor
  - **0.5 puncte** pentru prăjiturile fără vanilie
  - **1 punct** pentru instrucțiunea finală
- **2 puncte** pentru b3
  - **1 punct** pentru grupare prăjitori, sumă calorii
  - **1 punct** pentru instrucțiunea finală

Subiect 3 (Sisteme de operare):

Oficiu **1**p

3.1 5p din care

(a) 1p: Se creeaza 6 (sau N) procese fiu

(b) 1p: Se va afisa valoarea 7 (sau N+1)

2p: Procesul parinte si fiii isi transmit in cerc un intreg pe care fiii il incrementeaza

(c) 1p: Intregul circula doar intre parinte si fiii 0, 2 si 4. Fiilor 1, 3 si 5 nu li se trimite nimic si sunt blocati in apelul sistem read. Parintele este blocat in apelul sistem wait.

3.2 4p din care

(a) 1p: Explicarea functionarii

(b) 1p: Se afiseaza procentul de erori “segmentation fault” din numarul total de erori

(c) 1p: Explicarea liniei 8

(d) 1p: Explicarea necesitatii “\”