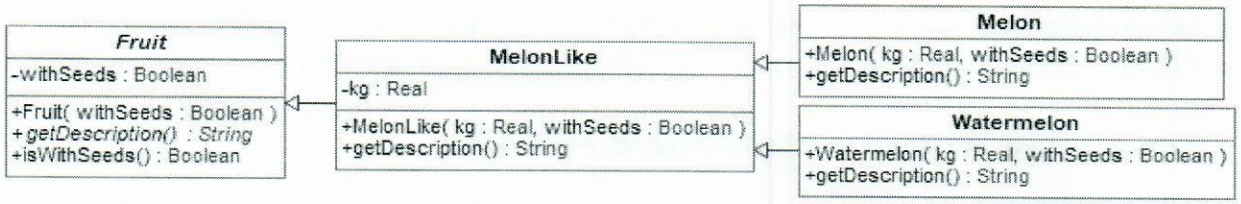
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2017 július 3
Informatika - magyar nyelv
2. VÁLTOZAT

1. tétel Algoritmusok és programozás

Írjunk egy programot a C++, Java, C#, vagy Python programozási nyelvek egyikében, mely:

a). **definiálja** a ***Fruit*** (gyümölcs), **MelonLike** (dinnye-szerű gyümölcs), **Melon** (sárgadinnye) és **WaterMelon** (görögdinnye) osztályokat a következő UML-diagram alapján:



- A *kg* érték szigorúan pozitív. A megszorításokat a konstruktorokban ellenőrizzük.
- A ***Fruit*** osztály absztrakt, a *getDescription()* (leírás) absztrakt metódus.
- A ***Fruit*** osztály **isWithSeeds()** függvénye igaz értéket térít vissza, ha a gyümölcs sokmagos, hamis értéket ellenkező esetben.
- A **MelonLike** osztály **getDescription()** metódusa egy karakterláncot térít vissza a következő struktúrával: előbb a súlyt, majd a „dinnye-szerű" szót, majd a „magos" vagy „magnélküli" szavakat, annak függvényében, hogy van-e magja vagy nincs. A **Melon** (sárgadinnye) és a **Watermelon** (görögdinnye) osztályok **getDescription()** metódusai az alaposztály-beli értékhez a „sárgadinnye", illetve a „görögdinnye" szavakat adják hozzá és térítik vissza.

b). **definiál** egy függvényt, mely visszatéríti egy *gyümölcs* beszúrási pozícióját egy ***Fruit*** típusú elemeket tartalmazó, a **getDescription()** függvény értéke szerint ábécé sorrendbe rendezett listába. Használjunk bináris keresést.

c). **definiál** egy függvényt, mely – a b.) alpontban megírt függvény használatával – beszúr egy gyümölcsöt – ***Fruit*** típusú elemet – egy ***Fruit*** típusú elemeket tartalmazó ábécé sorrendbe rendezett listába, a rendezési kulcs itt is a **getDescription()** értéke.

d). **definiál** egy függvényt, melynek két bemenő paramétere a *withSeeds* (magos vagy sem) logikai változó és egy ***Fruit*** elemeket tartalmazó lista. A függvény megjeleníti (kiírja) azon elemeket, melyek magosak/nem magosak, a *withSeeds* paraméter értékétől függően.

e). **definiálja** a program fő-függvényét, mely (1) léterhoz egy listát a következő tartalommal: egy 6 kg-os, magnélküli görögdinnye (**Watermelon**), egy 10 kg-os, magos sárgadinnye (**Melon**), egy 11 kg-os, magnélküli dinnye-szerű gyümölcs (**MelonLike**), illetve egy 13 kg-os, magos görögdinnye (**Watermelon**). Szúrjunk be a felépített listába – a c.) pontban definiált függvényt használva – egy 12 kg-os, magnélküli görögdinnyét (**Watermelon**), majd a d.) alpontbeli függvényt használva írassuk ki rendre a magos majd a magnélküli gyümölcsök listáját.

f). Adjuk meg a **Lista** adattípusra a használt műveletek specifikációját.

Megjegyzés:

- **Adjuk meg a használt programozási nyelvet.**
- **Ne használjunk rendezett konténereket és létező rendezési műveleteket.**
- **Ne használjunk a specifikáción kívüli függvényeket.**

*Használhatunk **adatstruktúrákat** tartalmazó függvénykönyvtárakat (C++, Java, C#, Python).*

2. tétel Adatbázisok

a.) Tervezzük meg a TIFF filmfesztivál infomációit tartalmazó **relációs adatbázis sémát,** melynek táblái **3NF**-ban vannak:

- **helyszínek**: kód, név, cím;
- **filmek**: cím, év, műfajok listája (ahol a műfajnak van kódja, neve, leírása), szereplők listája (ahol a szereplőnek van kódja, neve) és vetítések listája (ahol a vetítésnek van kódja, helyszín kódja, dátuma, órája);
- **eladott jegyek**: vetítés kód, sor száma, ülőhely száma.

A funkcionális függőségeket megadva, indokoljuk, hogy a táblák **3NF**-ben vannak.

b.) Adjuk meg az **a.** pont adatbázisára vonatkozóan, SQL parancsot **vagy** relációs algebrát használva:

i) azon **helyszíneket** (név, cím), ahol vetítettek legalább egy *komédiát* és legalább egy *drámát* is.

ii) azon filmekre eladott **jegyek számát**, amelyekben szerepel *Alain Delon* és a *Főtér* helyszínen voltak vetítve.

iii) azon **film(ek)et** (cím, év), ami(k)re a legtöbb jegyet vásárolták.

3. tétel Operációs rendszerek

3.1 Feltételezve, hogy az alábbi kódrészlet minden utasítása hiba nélkül hajtódik végre, válaszoljunk a következő kérdésekre:

```
1  int main(){
2    int p[2], i=0;
3    char c, s[20];
4    pipe(p);
5    if (fork()==0){
6      close(p[1]);
7      while(read(p[0], &c, sizeof(char))){
8        if( i < 5 || i > 8 ){
9          printf("%c", c);
10       }
11       i++;
12     }
13     printf("\n"); close(p[0]);
14     exit(0);
15   }
16   printf("Result: \n");
17   strcpy(s,"exam not passed");
18   close(p[0]);
19   write(p[1], s, strlen(s)*sizeof(char));
20   close(p[1]);
21   wait(NULL);
22   return 0;
23 }
```

a) Rajzoljuk le a létrejött folyamatok hierarchiáját, a szülő folyamatot is beleértve.

b) Adjuk meg a program által kiírt összes sort, az azokat kiíró folyamat specifikálásával.

c) Hány karaktert olvasunk ki a pipe-ból?

d) Milyen hatással van a folyamatok befejeződésére, ha hiányzik a 20. sor?

e) Milyen hatással van a folyamatok befejeződésére, ha hiányzik a 20. és 21. sor?

3.2 Az alábbi UNIX Shell script egy futtatását tekintve, válaszoljunk a következő kérdésekre.

```
1  f=`find . -type f`
2  d=`find . -type d`
3
4  for x in $f; do
5    for y in $d; do
6      if [ $x = $y ]; then
7        echo "OK"
8      fi
9    done
10 done
```

a) Hányszor lesz kiírva az „OK”? Indokoljuk a választ.

b) Mi az `f` változó értéke?

c) Mi a `d` változó értéke?

d) Mi az `x` változó értéke?

e) Mi az `y` változó értéke?

Megjegyzések.

- Mindegyik tétel kötelező; a tételeknél teljes megoldásokat kérünk.
- A dolgozat minimális átmenő jegye az ötös (5,00).
- Munkaidő: három (3) óra.

BAREM INFORMATICĂ

Subiect 1 (Algoritmică și Programare):

Oficiu – **1**p

Definirea claselor *Fruit* și *MelonLike*– **1**p din care
- relația de moștenire – 0.25
- atribute – 0.25
- constructor – 0.25
- metode - 0.25

Definirea claselor *Watermelon* și *Melon*– **1**p din care
- relația de moștenire – 0.25
- constructor – 0.25
- metode – 0.5

Funcția de la punctul b) – **2**p din care
- signatura corectă - 0.1p
- algoritmul de căutare binară - 1.8p
- returnare rezultat - 0.1p

Funcția de la punctul c) - **2**p din care
- signatura corectă - 0.1p
- determinare poziție de inserare - cu fct de la b) – 0.2p
- inserarea elementului pe poziția determinată anterior – 1.7p

Funcția de la punctul d) – **1**p din care
- signatura corectă - 0.1p
- parcurgere listă – 0.4p
- verificare condiție – 0.1p
- accesare și tipărire element – 0.4p

Funcția principală e) – **0.5p**

f) Specificațiile operațiilor folosite pentru tipul de dată **Listă**– **1.5**p

Subiect 2 (Baze de date)

1 punct oficiu

Problema a:
- **2 puncte** pentru tabelele în 3NF
- **2 puncte** pentru justificare:
  - **1 punct** definiții
  - **1 punct** explicații

Problema b:
- **1.5 puncte** pentru i
- **1 punct** pentru ii
- **2.5 puncte** pentru iii

Subiect 3 (Sisteme de operare):

Oficiu – 1p

3.1 – 5p din care
- a) Diagrama ierarhiei - 1p
- b) Linia părintelui – 0.5p
  Linia fiului – 0.5p
- c) 15 caractere – 1p
- d) Niciun proces nu se termină. Fiul blocat la read, părintele la wait – 1p
- e) Procesele se termină, pipe-ul fiind închis la terminarea părintelui – 1p

3.2 – 4p din care
- a) Nu se afișează nimic – 1p
  Justificare – 1p
- b) Numele și calea tuturor fișierelor normale din directorul curent și toate subdirectoarele – 0.5p
- c) Numele și calea tuturor directoarelor din directorul curent și toate subdirectoarele – 0.5p
- d) Numele și calea fiecărui fișier lista din $f – 0.5p
- e) Numele și calea fiecărui director din lista $d – 0.5p