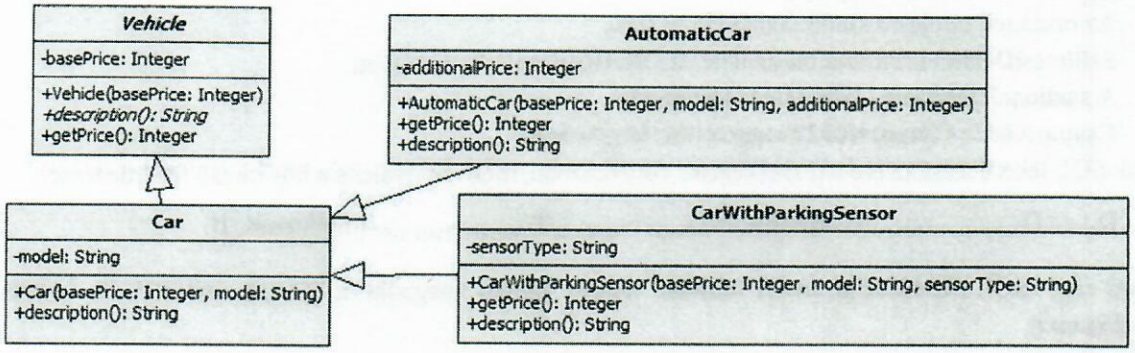
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2018 július 2
Informatika - magyar nyelv
1. VÁLTOZAT

1. tétel Algoritmusok és programozás

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

a) **Definiálja** a következő osztályokat: ***Vehicle*** (absztrakt), ***Car***, ***AutomaticCar***, illetve ***CarWithParkingSensor***, az alábbi UML-diagram alapján:



- a *basePrice* (alapár) és az *additionalPrice* (felár) mezők szigorúan pozitívak, a *sensorType* (érzékelő-típus) és a *model* (autó-modell) mezők nem null-értékűek és nem üresek. A megszorításokat a konstruktorokban ellenőrizzük.
- A ***Vehicle*** osztály **getPrice()** metódusa a jármű alapárát, az ***AutomaticCar*** osztály **getPrice()** metódusa az alapár és a felár összegét téríti vissza, a ***CarWithParkingSensor*** osztály **getPrice()** metódusa pedig a jármű alapáránál 2500 egységgel nagyobb értéket térít vissza.
- A ***Car*** osztály **description()** metódusa az autó modelljét, az **AutomaticCar** osztály **description()** metódusa az "Automatic car " szöveg és az autó modelljének az összefűzését, és a **CarWithParkingSensor** osztály **description()** metódusa pedig a "Car with parking sensor " szövegnek, valamint a szenzor *típusá*nak, egy szóköznek és az autó modelljének az összefűzését téríti vissza.

b) **Definiál** egy **függvényt**, melynek bemenő paramétere egy L lista, ***Vehicle*** típusú elemekkel; a függvény egy <*model*, *nrCars*> párosok listáját téríti vissza, ahol *nrCars* az L listában található minden egyes modellhez tartalmazza a hozzá tartozó autók számát. Az eredmény-listában az L listában található modellek közül mindegyik *model* csak egyszer szerepel.

c) **Definiál** egy **függvényt**, melynek bemenő paramétere egy L lista, ***Vehicle*** típusú elemekkel; átrendezi a listát úgy, hogy az [1000, 2000] közötti autó-árral rendelkező autók az 1000-nél kisebb vagy 4000-nél nagyobb árú autók elé kerüljenek; a függvény megőrzi az objektumok eredeti sorrendjét. Ne használjunk segédstruktúrákat.

d) **Definiál** egy **függvényt**, mely egy bemenő - ***Vehicle*** típusú - listára, kiírja az autók leírását (description).

e) **Definiálja** a program fő-függvényét, mely létrehozza a következő autó-listát (a nem pontosított mezőknek adjunk értéket): egy Audi, egy automata sebességváltós Audi, egy Toyota, egy automata sebességváltós Mercedes, illetve egy parkolószenzoros Opel. A megépített listára hívjuk meg a b) pontnál megírt függvényt és jelenítsük meg a visszatérített listát. Hívjuk meg a c) pontban megírt függvényt a listára, majd a d) pont segítségével írjuk ki az eredményt.

f) A programban szereplő **Lista** adattípusra adjuk meg a a használt műveletek specifikációját.

Megjegyzés:

1. **Adjuk meg a használt programozási nyelvet.**
2. **Ne használjunk a specifikáción kívüli metódusokat.**
3. **Ne használjunk rendezett konténereket és predefiniált rendezési műveleteket.**

*Az **adattípusokhoz** használhatunk létező függvénykönyvtárakat (C++, Java, C#, Python).*

2. tétel Adatbázisok

Legyen egy adatbázis, mely a nemzeti labdarúgó válogatottak mérkőzéseinek eredményeit tárolja (egy csapat egy dátumon egy mérkőzést játszhat). Az adatbázis szerkezete a következő:

- ***Csapatok*** tábla, attribútumai: **CsKód, Ország, Kontinens**;

- *Játékosok* tábla, attribútumai: **JKód, CsapatKód, Név, ÉletKor, MeccsekSzáma**;
- *Meccsek* tábla, attribútumai: **CsapatKód1, Gólok1, CsapatKód2, Gólok2, Dátum, Stadion, HelyekSzáma, Város, Ország**.

1. Határozzuk meg az elsődleges és külső kulcsokat!
2. Adjunk meg legalább 4 funkcionális függőséget, melyek nem kód attribútumokra vonatkoznak.
3. A következő szerkezeti módosítások közül melyek szükségesek ahhoz, hogy az adatbázis harmadik normál formában (**3NF**) legyen. Indokoljuk a választást!
   a) Az országok tárolására külön tábla létrehozása.
   b) **SzületésiDátum** attribútum használata az **ÉletKor** attribútum helyett.
   c) A stadionok tárolására külön tábla létrehozása.
   d) **CsapatKód1** > **CsapatKód2** megszorítás létrehozása.
4. Írjunk SQL lekérdezést az eredeti szerkezetre vonatkozóan, mely ekvivalens a következő lekérdezéssel:

$$\Pi_{\text{Név}} (\sigma_{\text{Kontinens = 'Ázsia'}} (\textbf{Csapatok} \bowtie_{\text{CsKód = CsapatKód}} \sigma_{\text{MeccsekSzáma >100}} (\textbf{Játékosok})))$$

5. Írjunk egy SQL lekérdezést, mely minden csapat esetén megadja a játszott mérkőzések számát. (***Ország, MeccsekSzáma***).

3. tétel Operációs rendszerek

3.1 Az alábbi kódrészlet sikeresen fordul le a `pr` futtatható állománnyá. Az `f` függvény `w` argumentuma megadja, hogy melyik FIFO-t fogjuk írásra használni: 0 az értéke, ha `a`-t illetve 1 ha `b`-t. Feltételezve, hogy minden utasítás hiba nélkül fut le, továbbá, hogy minden FIFO törölve van, majd újra létrehozva, minden egyes futtatás előtt, illetve, hogy `pr` az egyetlen program, mely hozzáfér ezekhez a FIFO állományokhoz, válaszoljunk az alábbi kérdésekre:

```
void f(char* a, char* b, int w, char* s) {
  int f[2], r=1-w; char c;
  if(fork() == 0) {
    f[0] = open(a, w==0 ? O_WRONLY : O_RDONLY);
    f[1] = open(b, w==1 ? O_WRONLY : O_RDONLY);
    write(f[w], s, 1);
    read(f[r], &c, 1);
    printf("%c\n", c);
    close(f[0]); close(f[1]);
    exit(0);
  }
}
int main(int n, char** a) {
  int i;
  for(i=1; i<n; i+=4) {
    f(a[i], a[i+1], a[i+2][0]-'0', a[i+3]);
  }
  for(i=1; i<n; i+=4) {wait(0);}
  return 0;
}
```

a) Készítsünk egy, a folyamatok hierarchiáját szemléltető, ábrát a `pr` egy olyan végrehajtására, amikor a kapott paramétereinek száma `4*K`.

b) Mit fog kiírni az alábbi végrehajtás? Indokoljuk a választ.

```
./pr p q 1 x
```

c) Mit fog kiírni az alábbi végrehajtás? Indokoljuk a választ.

```
./pr p q 1 x p q 0 y
```

d) Készítsünk egy ábrát, mely a gyerekfolyamatokat ábrázolja, illetve az általuk, a FIFO állományokon végrehajtott read/write műveleteket, a (c) pontban megadott végrehajtás esetén.

e) Mit fog kiírni az alábbi végrehajtás? Indokoljuk a választ.

```
./pr p q 1 x q p 1 y
```

3.2 A `sed "s/A/B/"` parancs kicseréli minden egyes sorban az `A` reguláris kifejezés első előfordulását a `B` karakterlánccal, miközben a `B`-ben található `\N` hivatkozás minden előfordulását kicseréli az `A`-ban zárójellel megjelölt `N`. kifejezésre. Mi lesz a kimenete az alábbi shell script-nek, amennyiben egy olyan katalógusban hajtjuk végre, mely C/C++ forráskódú állományokat, illetve fejlécállományokat tartalmaz? Magyarázzuk el részletesen a 3. sort: a parancsokat, ezek argumentumait, illetve a pipe-ot.

```
1 for F in *.c *.cpp *.h; do
2   if [ -f $F ]; then
3     grep "#include.*<" $F | sed "s/^.*<\(.*\)>.*$/\1/"
4   fi
5 done | sort
```

Megjegyzések.

- Mindegyik tétel kötelező; a tételeknél teljes megoldásokat kérünk.

BAREM INFORMATICĂ
VARIANTA 1

Subiect 1 (Algoritmică și Programare):

Oficiu – 1p
Definirea clasei abstracte Vehicle – 0.3p din care
- atribut – 0.1
- metode - 0.2

Definirea clasei Car – 0.3p din care
- relația de moștenire – 0.1
- atribut – 0.1
- metode - 0.1

Definirea clasei AutomaticCar– 1p din care
- relația de moștenire – 0.15
- atribut – 0.15
- constructor – 0.35
- metode – 0.35

Definirea clasei CarWithParkingSensor – 1p din care
- relația de moștenire – 0.15
- atribut – 0.15
- constructor – 0.35
- metode – 0.35

Funcția de la punctul b) – 2.25p din care
- signatura corectă - 0.1p
- construire listă perechi conținând modele distincte- 2p
- returnare listă rezultat – 0.15

Funcția de la punctul c) – 2.25p din care
- signatura corectă - 0.1p
- rearanjare listă – 2.15p

Funcția de la punctul d) – 0.4p din care
- signatura corectă - 0.1p
- afișare descrierei mașini din listă – 0.3p

Funcția principală e) – 0.5p
f) Specificațiile operațiilor folosite pentru tipul de dată Listă– 1p

Subiect 2 (Baze de date)

1. 0.5p (chei primare) + 0.5p (chei externe) = 1p
2. 0.25p x 4 = 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 3 (Sisteme de operare):

3.1.a - diagrama cu un părinte și mulți fii - 0.5p
- K procese fiu - 0.5p

3.1.b - nimic - 0.5p
- se blochează open la deschiderea FIFO-ului - 0.5p

3.1.c - x si y - 0.5p
- în ordine nedeterminabilă - 0.5p

3.1.d - diagrama circulară cu două procese și două FIFO-uri - 1p

3.1.e - nimic - 0.5p
- se blochează open-urile și creează deadlock - 0.5p

3.2 Rezultate - lista sortată a fișierelor header sistem incluse în surse - 1p

3.2 Linia 3 - grep: extrage liniile care includ fișiere header sistem - 0.5p
- grep: detaliere expresie regulară - 0.5p
- pipe: transmite output-ul lui grep ca input lui sed - 0.5p
- sed: pastrează pe linie doar numele fișierului header - 1p
- sed: detaliere comandă și expresie regulară – 0.5p