BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA ÉS INFORMATIKA SZAK

Licenszvizsga, 2019 szeptember 4
Informatika - magyar nyelv

1. VÁLTOZAT

Megjegyzések:

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

Operációs Rendszerek Tétel

1. Feltételezve, hogy minden szükséges header állomány benne van az alábbi programban, és minden utasítás helyesen lefut, válaszoljunk a következő kérdésekre:

```
1  int n=0, a[2], b[2];
2  void w(int p[2], char c) {
3    n++;
4    if(fork() == 0) {
5      close(p[0]); write(p[1], &c, 1); close(p[1]);
6      exit(0);
7    }
8  }
9  void r(int p[2]) {
10   char c;
11   n++;
12   if(fork() == 0) {
13     close(p[1]);
14     if(read(p[0], &c, 1) > 0) {printf("%c\n", c);}
15     close(p[0]);
16     exit(0);
17   }
18 }
19 int main(int argc, char** argv) {
20   pipe(a); pipe(b);
21   w(a, 'x'); w(a, 'y'); r(a); r(a);
22   close(a[0]);close(a[1]);close(b[0]);close(b[1]);
23   for(int i=0; i<n; i++) {wait(0);}
24   printf("%d\n", n);
25   return 0;
26 }
```

a) Rajzoljuk le a létrehozott folyamat-hierarchiát, a szülő-folyamatot is beszámítva.

b) Mit ír a képernyőre a program végrehajtáskor?

c) Mit ír a képernyőre a program végrehajtáskor, ha a 21. sort a következőre cseréljük:

```
w(a, 'x'); w(b, 'y'); r(a); r(b);
```

d) Mit ír a képernyőre a program végrehajtáskor, ha a 21. sort a következőre cseréljük:

```
w(a, 'x'); w(a, 'y'); r(a); r(b);
```

e) Magyarázzuk és indokoljuk meg a program működését, ha az 21. sort a d) pontnak megfelelően kicseréljük, és emellett töröljük a 22. sort.

2 Válaszoljunk a következő, az alábbi Unix shell-szkripthez kapcsolódó kérdésekre!

```
1  #!/bin/bash
2  for F in *.sh; do
3    A=`grep "^[ \t]*[^ \t#]" $F | wc -l`
4    B=`wc -l < $F`
5    if [ $A -lt `expr $B - $A` ]; then
6      echo $F
7    fi
8  done
```

a) Mit ír a képernyőre a szkript végrehajtáskor?

b) Mit ír a képernyőre a szkript végrehajtáskor, ha abból a katalógusból futtatjuk, amelyben csak maga a szkript található?

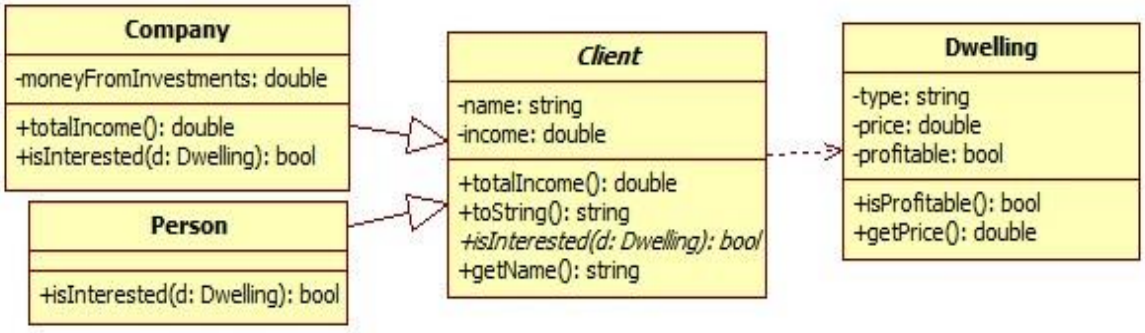
c) Magyarázzuk meg részletesen a 3. sorban levő reguláris kifejezést!

1. VÁLTOZAT

Algoritmusok és programozás TÉTEL

Írjunk egy programot a C++, Java, C#, vagy Python programnyelvek egyikében az alábbi követelmények szerint:

a) Definiálja a **Client** (Kliens), **Person** (Személy), **Company** (Cég), **Dwelling** (Lakás) osztályokat az alábbi UML-diagram alapján (a diagramban a konstruktorokat nem jelöltük). Az ábrán definiált metódusok közül csak a b) pontban jelölteket kell implementálni; a többit csupán definiáljuk.



- A **Client** osztály *name* (kliens név) attribútuma legkevesebb 3 hosszúságú, illetve az *income* (jövedelem) attribútuma szigorúan pozitív. A megszorításokat a konstruktorok implementálják.
- Az absztrakt **Client** osztálynak egy absztrakt **isInterested** metódusa van.
- Egy cég teljes jövedelme a cég jövedelme (*income*), melyhez hozzáadódik a befektetésekből származó összeg (*moneyFromInvestments*). Az ettől különböző kliensek teljes jövedelme a jövedelem attribútum értékével egyenlő.
- Egy személy érdeklődik egy lakás (**Dwelling**) iránt, ha a lakás árának a 360-ad része kisebb vagy egyenlő, mint a személy teljes jövedelmének a fele. Egy cég érdeklődik egy lakás iránt, ha a lakás árának a 12-ed része kisebb a cég teljes jövedelménél és ha a lakás jövedelmező.
- A **toString** metódus visszatéríti a nevet, melyhez hozzáfűzzük a teljes jövedelmet a cég és a személy számára egyaránt.

b) Implementáljuk a következő metódusokat az a) pont UML-ábrájából: a **Client**, **Person**, **Company**, **Dwelling** osztályok konstruktorait; a **Client** és **Company** osztályok **totalIncome** metódusait, illetve az **isInterested** metódust a **Person** és **Company** osztályoknál.

c) Definiáljunk egy függvényt, mely egy **kliens-listát** és egy rendezési relációt kap (melyet egy boolean logikai kimenetű, klienseket összehasonlító függvénnyel adunk meg) és rendezi a listát a függvény által definiált rendezés szerint. A rendezést egy $\theta(n \cdot log_2 n)$ bonyolultságú algoritmussal implementáljuk.

d) Definiál egy függvényt, mely egy szótárban tárolt **klienseket** (ahol az elem kulcsa a kliens neve, az értéke a kliens objektum) valamint egy lakást kap bemenetként, és visszatéríti - a c)-ben megírt függvény segítségével – az összes, a lakás iránt érdeklődő kliens listáját, mely lista a név attribútum szerint növekvő sorrendben rendezett.

e) Létrehoz a program fő-függvényében (main) egy kliens-értékű szótárt és beszúrja a következő klienseket (a nem specifikált mezőknek adjatok tetszőleges értéket): két céget, egyiknek a neve legyen “TwoStar” a másiknak a neve legyen “Fort”, illetve két személyt, az egyik “Ana”, a másik “Mihai”. Hozzunk létre két lakás típusú objektumot: az egyik típusa “panellakás”, ára 150,000 és nem jövedelmező; a második típusa “ház”, ára 500,000 és jövedelmező. A meglévő kliens-szótárra és mindkét lakásra hívjuk meg a c) pontban definiált függvényt, majd írassuk ki az eredménylistákat a **toString** metódus használatával.

f) Adjuk meg a **Szótár (Dictionary)** adatstruktúra **hozzáadás** és **keresés** metódusainak a specifikációját.

- **Specifikáljuk a használt programozási nyelvet!**
- **Ne használjunk a specifikáción kívüli metódusokat (a konstruktorok kivételével)!**
- **Nem szabad rendezett konténereket és előredefiniált rendező műveleteket használni!**

*Az **adattípusokra** használhatunk létező könyvtári függvényeket (Python, C++, Java, C#).*

1. VÁLTOZAT

Adatbázisok TÉTEL

Tekintsük az alábbi adatbázist, mely egy népszavazásra készült elektronikus szavazó alkalmazás adatait tartalmazza. Az adatbázis szerkezete a következő:

- ***Személyek*** tábla, attribútumai: **SzemKód**, **Név**, **CNP**, **HelységKód**, **MegyeKód**, **RégióKód, SzavazottE**;
- ***Szavazatok*** tábla, attribútumai: **SzavKód**, **Dátum, Óra**;
- ***Kérdések*** tábla, attribútumai: **KérdésKód**, **KérdésSzövege, KategóriaMegnevezés**, **KategóriaLeírás;**
- ***Válaszok*** tábla, attribútumai: **SzavKód**, **KérdésKód**, **IgenVálasz, NemVálasz, Érvénytelen.**

A rendszerben egy kérdésre adott választ akkor tekintünk érvénytelennek, ha az *Igen* és a *Nem* közül egyik sem lett kiválasztva, vagy ha mindkettő ki lett választva. Az **IgenVálasz, NemVálasz, Érvénytelen** és **SzavazottE** attribútumok csak a 0 és 1 értékeket vehetik fel.

1. Határozzuk meg minden tábla esetén a kulcsjelölteket és a külső kulcsokat!

2. Adjunk meg legalább 4 funkcionális függőséget, melyek NEM a kulcsjelöltekre vonatkoznak!

3. Az alábbi szerkezeti módosítások közül melyek szükségesek ahhoz, hogy az adatbázis harmadik normálformában **(3NF)** legyen. Indokoljuk meg választásainkat!
   a) Külön tábla létrehozása a kategóriák tárolására.
   b) Az **Érvénytelen** attribútum törlése a ***Válaszok*** táblából.
   c) A ***Szavazatok*** tábla esetén a szavazás időpontjának (dátum és óra) egyetlen attribútumban való tárolása.
   d) A **SzemélyKód** attribútum hozzáadása a ***Szavazatok*** táblához, mellyel a **Személyek** táblára hivatkozhatunk.

4. Írjunk egy SQL lekérdezést, az eredeti szerkezetre vonatkozóan, mely ekvivalens az alábbi lekérdezéssel:

$$\Pi_{\text{KérdésSzövege,Óra}} \left(\sigma_{\text{Óra} > 18:00} \left(\textbf{\textit{Szavazatok}} \bowtie_{\text{Szavazatok.SzavKód=Válaszok.SzavKód}} \textbf{\textit{Válaszok}} \right. \right.$$
$$\left. \left. \bowtie_{\text{Kérdések.KérdésKód=Válaszok.KérdésKód}} \sigma_{\text{KategóriaMegnevezés='EU'}} (\textbf{\textit{Kérdések}}) \right) \right).$$

5. Írjunk egy SQL lekérdezést, az eredeti szerkezetre vonatkozóan, mely megadja minden kérdésre a válaszok összszámát, az érvényes *Igen* válaszok számát, az érvényes *Nem* válaszok számát és az érvénytelen válaszok számát (***KérdésSzövege, IgenekSzma, NemekSzáma, ÉrvénytelenekSzáma***).

BAREM INFORMATICĂ
VARIANTA 1

Subiect (Algoritmică şi Programare)

Oficiu – 1p
Definirea clasei abstracte Client – 0.6p din care
- atribute – 0.1
- constructor si metode – 0.5

Definirea clasei Company– 1.5p din care
- relația de moștenire – 0.2
- atribut – 0.1
- constructor – 0.4
- metode – 0.8

Definirea clasei Person – 0.7p din care
- relația de moștenire – 0.2
- constructor – 0.2
- metoda – 0.3

Definirea clasei Dwelling – 0.2p din care
- atribute– 0.1
- constructor - 0.1

Funcția de la punctul c) – 3p din care
- signatura corectă - 0.4p
- sortare in $\theta(n \cdot log_2 n)$ - 2.5p
- returnare listă rezultat – 0.1

Funcția de la punctul d) – 1.5p din care
- signatura corectă - 0.1p
- construire lista clienti in ordine alfabetica dupa nume – 1.3p
- returnare listă rezultat – 0.1

Funcția principală e) – 0.5p
f) Specificaţiile operaţiilor **adăugare** și **căutare** pentru tipul de dată **Dicționar** – 1p

Subiect Baze de date

1. 0.5p (chei candidat) + 0.5p (chei externe) = 1p
2. 0.25p x 4 = 1p
3. a, b
   - 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 procesul părinte având patru procese fiu – 1p
1.b Va afișa “x y 4” sau “y x 4” – 1p
1.c Identic cu punctul 3.1.b – 1p
1.d Va afișa “x 4” sau “y 4” – 1p
1.e Va afișa “x” sau “y” și se va bloca la linia 14 pentru că pipe-ul este deschis pentru scriere – 1.5p

2.a Numele scripturilor Shell din directorul curent cu mai puține linii de cod decât restul liniilor din script – 1p
2.b Nu va afișa nimic – 1p
2.c Inceput de linie, 0 sau mai multe spații sau taburi, orice caracter care nu e spațiu, tab sau diez – 1.5p