

ÉRETTSÉGI VIZSGA • 2011. május 13.

INFORMATIKAI ALAPISMERETEK

EMELT SZINTŰ ÍRÁSBELI ÉRETTSÉGI VIZSGA

JAVÍTÁSI-ÉRTÉKELÉSI ÚTMUTATÓ

**NEMZETI ERŐFORRÁS
MINISZTERIUM**

Fontos tudnivalók

I. rész

Általános megjegyzések:

- Ha a vizsgázó nem a kérdésben meghatározottak szerint válaszol, akkor a válasz nem fogadható el! (Pl.: **H** betű helyett nem válaszolhat **N** betűvel.)
- A feleletválasztásos tesztfeladatnál javítani tilos, a javított válaszok nem értékelhetők!
- Ha egy kérdésre a jó válasz(ok) mellett a vizsgázó válaszában hibás választ is megjelöl, akkor a kérdésre adható pontszámból le kell vonni a rossz válaszok számát. Negatív pontszám nem adható, ezért több hibás válasz esetén a minimális pontszám nullánál kevesebb nem lehet.
Pl.: Ha egy jó válasz mellett a vizsgázó egy hibás választ is bejelöl, akkor 0 pontot kell adni. Ez nem vonatkozik azokra a kérdésekre, ahol a **(minden helyes részválasz 1 pont)** szöveg szerepel.
- A kifejtős kérdések (nem feleletválasztós) válaszainál nem a szó szerinti, hanem a helyes tartalmi, illetve a lényegi válaszok megadását kell értékelni. Ha a vizsgázó válaszában a tartalmi vonatkozásai megfelelnek a megoldási útmutatóban megadott válasznak, akkor a válasza adható pontot meg kell adni. Ha csak kis részben, vagy pedig nem felel meg a kapott válasz, akkor pont nem jár a válaszáért.
- A pontszámok az **I.** részben a megadott részletezésnél tovább nem bonthatók (0,5 pont nem adható)
- Egyes esetekben előfordulhat, hogy egy általánostól eltérő rendszer használata miatt valamely kérdésre a vizsgázó nem a várt válasz adja, de *a válasza és az indoklása elfogadható*. Ilyen esetben a kérdésre adható pontszámot meg kell adni.
Pl.: Táblázatkezelőkben magyar beállításnál a tizedesek elválasztásának a jele a **vessző**, és ez a várt válasz. Ha a vizsgázók munkájuk során angol beállítást használnak, vagy a vizsgázó odaírja ezt megjegyzésként, akkor az előző helyett az angol beállítású környezetben használt **pont** lesz a helyes válasz.

A javítási-értékelési útmutatóban feltüntetett válaszokra kizárólag a megadott pontszámok adhatók.

A megadott pontszámok további bontása csak ott lehetséges, ahol erre külön utalás van. Az így kialakult pontszámok csak egész pontok lehetnek.

I. Egyszerű rövid, illetve kifejtendő szöveges választ igénylő írásbeli feladatok**Hardver**

1)	d	$19\,200 / (1 + 8 + 1) = 1\,920$	3 pont
2)	a	$1\,280 * 720 * 2 = 1\,843\,200$ bájt	3 pont
3)	c	bit/s	1 pont
4)	b	A processzor által ismert műveletek és utasítások összessége.	1 pont
5)	c	A csatlakozófej különbözik.	1 pont
	<i>vagy</i> d	A D-SUB analóg, a DVI digitális jelet visz. Megjegyzés: c és d együttes jelölése is elfogadható, de a feladatra maximum 1 pont adható.	
6)	I, H, H, H, H		5 pont
7)	Webkamera		2 pont
8)	CCD: c ADSL: a CYMK: d AGP: b		2 pont
9)	b	Az USB legfrissebb verziója a 2.0-ás.	1 pont

Szoftver

10)	3-2-1-5-4		2 pont
11)	I, I, I, H, I, H		6 pont
12)	b	a processzor által aktívan használt, gyors elérésű memória, amely a futó programokat és azok adatait tárolja.	2 pont
13)	c	copy ment.txt+proba.txt p.txt	3 pont

Szövegszerkesztés, táblázatkezelés

14)	H, H, H, I, H, H		6 pont
15)	b	Kijelölöm a szövegrészt, majd a hasábok számát 2-re változtatom.	2 pont

Informatikai alapok

16)	a A bináris számok könnyen átalakíthatók 16-os számrendszerbe, és itt kevesebb számjeggyel írhatjuk le ugyanazokat az értékeket	1 pont
17)	$F=((A*C)+(B*D))*E$, másképpen: $F=((A \text{ and } C) \text{ or } (B \text{ and } D)) \text{ and } E$	1 pont
18)	<p>I.... Ha az A, B, C, D logikai változók értéke rendre 1,0,0,1, és</p> <p>E=(A AND (NOT (B AND C)))OR(D OR (NOT (A AND C)))</p> <p>akkor az E logikai változó értéke 1 lesz.</p> <p>H... Az A=01100011 és a B= 00110010 bitsorozat azonos helyi értékű bitjei között kizáró vagy műveletet végezve az eredmény bitsorozat decimális értéke 123 lesz.</p> <div><div>01100011</div><div>00110010</div><div>01010001 ez decimálisan 81</div></div> <p>I.... Ha az A, B, C logikai változók értéke rendre 0,1,1, és</p> <p>E=A OR (NOT (B AND NOT(C)))</p> <p>akkor az E logikai változó értéke 1 lesz.</p>	3 pont
19)	c Decimálisan: 2035, binárisan 11111110011	1 pont

Hálózati alapismeretek, HTML

20)	b 11000110 00101011 10001011 00111001	2 pont
21)	c <center> Ez az első <u>próba</u></center>	1 pont
22)	c Szállítási réteg.	1 pont
	Összesen:	50 pont

II. Programozási feladatok számítógépes megoldása

1. feladat

10 pont

Feladatkitűzés:

Írjon programot, amely előállítja egy pozitív egész szám római szám alakját az alábbiak szerint!

- A program a felhasználótól kérje be a pozitív egész számot!
- A beolvasás során semmilyen ellenőrzést nem kell végezni, feltételezzük, hogy a szám az [1..3999] egész intervallumba esik.
- Az átváltás szabályai a következő táblázatban foglalhatók össze:

	0	1	2	3	4	5	6	7	8	9
Egyesek	-	I	II	III	IV	V	VI	VII	VIII	IX
Tízesek	-	X	XX	XXX	XL	L	LX	LXX	LXXX	XC
Százások	-	C	CC	CCC	CD	D	DC	DCC	DCCC	CM
Ezresek	-	M	MM	MMM	-	-	-	-	-	-

- A különböző helyi értékeket külön-külön át kell váltanunk, és egyszerűen egymás mellé kell írunk.
- Példák:

Szám	Ezresek	Százások	Tízesek	Egyesek	Római
72	-	-	LXX	II	LXXII
953	-	CM	L	III	CMLIII
2618	MM	DC	X	VIII	MMDCXVIII

Fontos megkötések a program írásakor:

- Az egyesek, tízesek, százások, ezresek esetében ugyanazon érték esetében ugyanaz a programrész végezze el az átváltást, csak a helyi értéktől függően más-más szimbólumokat használjon!
 - Például: egy 7-es érték az egyesek esetén VII, a tízesek esetén LXX, a százások esetén DCC alakú lesz, amelyek az azonos szerkezet miatt ugyanazzal a programrésszel előállíthatók.
 - Segítség: érdemes egy függvényt írnia, amelynek paraméterei a helyi érték, és az átváltandó érték.
- Ahol csak lehet, alkalmazzon ciklust az elágazások kiküszöbölésére!
 - Például: a 5,6,7,8 értékek ugyanazzal a ciklussal átválthatók.
- Ügyeljen arra, hogy a 0 érték átalakításakor a program üres stringet állítson elő!

Beadandó a feladatot megoldó program forráskódja.

A feladat megoldásaként teljes, fordítható és futtatható kódot kérünk, mely az adatokat billentyűzetről (standard input) olvassa, és a képernyőre (standard output) írja ki. Vizuális fejlesztőeszköz használata esetén a megoldást konzol (szöveges ablakban futó) alkalmazásként kérjük elkészíteni.

Mintamegoldás: a program C# nyelven kódolva

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Feladat1
{
    class Romai
    {
        public static char[] romaiszamjegyek = {'I','V','X','L','C','D','M'};
        public static string arabszamjegybols_romai(int ertekek, int helyiertekek)
        {
            string r = "" ;
            if (!(ertekek == 4 || ertekek == 9)) // Ha ertekek nem 4, vagy 9
            {
                if (ertekek >= 5) // Beállítás V-re, L-re, vagy D-re
                {
                    r += romaiszamjegyek[2 * helyiertekek+1];
                }
                for (int i = 0; i < ertekek%5; i++)
                    //A megfelelő számú I, X vagy C mögé fűzése
                {
                    r += romaiszamjegyek[2 * helyiertekek];
                }
            }
            else // Ha az ertekek 4 vagy 9, akkor
                r += String.Concat(
                    romaiszamjegyek[2 * helyiertekek],
                    romaiszamjegyek[2 * helyiertekek + 1+(ertekek/5)]);
            return r;
        }

        public static string romai(int arab)
        {
            string s = arab.ToString();
            string romai = "";
            for (int i = 0; i < s.Length ; i++)
            {
                romai +=
                    arabszamjegybols_romai(s[i]-48, s.Length-i-1).ToString();
            }
            return romai;
        }
    }

    class teszt
    {
        public teszt()
        {
            Console.WriteLine("Római szám\n");
            Console.WriteLine("Adjon meg egy egész számot 1 és 3999 között: ");
            int arabszam = int.Parse(Console.ReadLine());
            Console.WriteLine("Római szám alakban: ");
            Console.WriteLine(Romai.romai(arabszam));
            Console.ReadLine();
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            teszt t = new teszt();
        }
    }
}
```

Értékelés:

- a) A programkód szintaktikailag hibátlan, lefordítható 1 pont
– Ez a pont csak abban az esetben adható meg, ha a programkód tartalmaz a b-d szakaszokba tartozó, összességében legalább 3 pontot érő részmegoldást.
- b) Az adat beolvasása, végeredmény kiírása 1 pont
– Egyértelmű beolvasás, a végeredmény kiírása
- c) Átalakítás római számmá..... 8 pont
– A 0,1,2,3 értékek helyes átváltása legalább egy helyi értéken: 1 pont
– Az 5,6,7,8 értékek helyes átváltása legalább egy helyi értéken: 1 pont
– Az átváltás az előző két esetben ciklussal történik: 1 pont
– A 0,1,2,3 és 5,6,7,8 értékek átváltása minden helyi értéken helyes: 1 pont
– A 4 és 9 értékek helyes átváltása legalább egy helyi értéken: 1 pont
– A 4 és 9 értékek átváltása minden helyi értéken helyes: 1 pont
– A számjegyek átváltott alakja összefűzésre kerül: 1 pont
– Az összefűzés a helyes sorrendben történik: 1 pont

2. feladat**10 pont****Feladatkitűzés:**

Egy tenisz adogatógép tesztelése során a teniszlabdákat különböző kezdősebességgel és szöggel lövik ki, majd lemérik, hogy milyen távolságra repülnek a labdák.

Írjon programot, amely lehetőséget ad arra, hogy meghatározhassuk a kipróbált kezdősebességekhez és kilövési szögekhez tartozó távolságokat, és egyszerű statisztikát készíthessünk a mért és számított távolságértékek közötti különbségekről!

– Adatbevitel

- A program tegye lehetővé az adatok bevitelét a billentyűzetről!
 - A kezdősebesség m/s-ban megadott pozitív valós szám, amelynek maximális értéke 40 m/s.
 - A kilövési szög fokokban megadott pozitív, de 90-nél kisebb egész érték.
 - A mért távolság m-ben megadott pozitív valós érték.
- A beolvasás során a program csak az előbb leírt feltételeknek megfelelő adatokat fogadjon el! Típusellenőrzést nem kell végeznie!
- A beolvasás akkor érjen véget, ha a felhasználó kezdősebességként 0 m/s-ot ad meg!

– Határozza meg a program, hogy mely bemenő paraméterek esetén volt a mért és számított értékek között a legnagyobb az abszolút eltérés!

- A távolság kiszámítására az $s = \frac{v_0^2 \cdot \sin 2\alpha}{g}$ képletet alkalmazzuk, ahol v_0 a

kezdősebesség, α a kilövési szög, s pedig távolság, g értéke $9,81 \frac{m}{s^2}$.

- A kiszámított távolságértéket ne kerekítsük!
- Az eltérést %-ban adjuk meg, 100%-nak a program által kiszámított távolságot tekintjük!
 - A %-os eredmény egészre legyen kerekítve!
 - Pl. ha a számított távolság 100 m, a mért távolság pedig 90 m, akkor az eltérés -10%, tehát az abszolút eltérés 10%.
 - Ha a számított távolság 10 m, a mért távolság 11,5 m, akkor az abszolút eltérés 15%.
- A vizsgálat elvégzése után a program írja ki, hogy hányadik tesztelés adatai esetén adódott a legnagyobb abszolút eltérés!

A feladat megoldásaként teljes, fordítható és futtatható kódot kérünk, mely az adatokat billentyűzetről (standard input) olvassa, és a képernyőre (standard output) írja ki. Vizuális fejlesztőeszköz használata esetén az algoritmust konzol alkalmazásként (szöveges ablakban futó) kérjük elkészíteni! Beadandó: a feladatot megoldó program forráskódja.

Mintamegoldás: a feladat egy lehetséges megoldása C# nyelven

(A tördelési problémák miatt a fájlban mellékelt megoldáshoz képest a nagyon hosszú sorok néhány helyen áttörve láthatók!)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Feladat2
{
    class Tenisz
    {
        private double v;
        private int alfa;
        private double sSzamitott, sMert;
        private int n;

        public Tenisz()
        {
            beolvas_feldolgoz();
        }

        public void beolvas_feldolgoz()
        {
            bool kilep = false;
            n = 0;
            int ind = -1;
            int maxelteres = 0;
            Console.WriteLine("=> Adatok beolvasása:");
            Console.WriteLine();
            do
            {
                Console.WriteLine((n + 1) + ". lövés: ");
                Console.WriteLine(" Kezdősebesség (m/s) ");
                do
                {
                    Console.Write("      >");
                    v = Double.Parse(Console.ReadLine());
                }
                while (!(v >= 0) && (v <= 40));
                kilep = (v == 0);
                if (!kilep)
                {
                    Console.WriteLine(" Szög (fok) ");
                    do
                    {
                        Console.Write("      >");
                        alfa = int.Parse(Console.ReadLine());
                    }
                    while (!(alfa > 0) && (alfa < 90));
                    Console.WriteLine(" Mért távolság (m) ");
                    do
                    {
                        Console.Write("      >");
                        sMert = Double.Parse(Console.ReadLine());
                    }
                    while (!(sMert > 0));

                    sSzamitott =
                        v * v * Math.Sin(Math.PI * 2 * alfa / 180) / 9.81;
                    int elteres =
                        (int)Math.Round(((Math.Abs(sMert / sSzamitott - 1)) * 100));
                }
            }
            while (kilep == false);
        }
    }
}
```

```
        Console.WriteLine("\n Számított távolság: "
                           +sSzamitott+" m");
        Console.WriteLine("    Eltérés           : " + elteres +
                           " %\n");

        if (elteres > maxelteres)
        {
            maxelteres = elteres;
            ind = n;
        }
        n++;

    }
}
while (!kilep);
if (n > 0)
{
    Console.WriteLine("=> Eredmény:");
    Console.WriteLine();
    Console.WriteLine("  A mért és a számított érték között a
                      legnagyobb eltérés a(z) " + (ind + 1) +
                      ". lövésnél volt." );}

    Console.ReadLine();
}
}

class Program
{
    static void Main(string[] args)
    {
        Tenisz t = new Tenisz();
    }
}
}
```

Értékelés:

- a) A programkód szintaktikailag hibátlan, lefordítható 1 pont
– Ez a pont csak abban az esetben adható meg, ha a programkód tartalmaz a b-d szakaszokba tartozó, összességében legalább 3 pontot érő részmegoldást.
- b) Adatok beolvasása 4 pont
– A kezdősebesség ellenőrzött beolvasása, valós érték adható meg: 1 pont
– A szög ellenőrzött beolvasása, egész érték adható meg: 1 pont
– A mért távolság ellenőrzött beolvasása, valós érték adható meg: 1 pont
A feladat megoldásához nem szükséges az értékeket tömbben tárolni.
– A fenti adatokat ismételten bekéri, az adatbekérés a 0 végjelig tart: 1 pont
- c) Számítások 4 pont
– Számított távolság meghatározása
 ○ Szög átváltása fokból radiánra: 1 pont
 ○ A képlet helyes kódolása: 1 pont
– %-os eltérés meghatározása: 2 pont
- d) A maximális eltérésű lövés sorszámának meghatározása, kiírása 1 pont

3. feladat**15 pont****Feladatkitűzés:**

Készítsen programot, amely egy osztály pénzügyi nyilvántartását modellezi a 2009/2010-es tanévre vonatkozóan!

- A tanulók létszáma 25 és 35 közötti legyen, a tanulókat a sorszámmal azonosítjuk.
- Minden tanulónak egyedi számlaegyenlege van, amely az év elején minimum 1000 Ft, maximum 3 500 Ft.
- Osztálypénz befizetés
 - A tanulónak szeptembertől júniusig havonta 1000 Ft osztálypénzt kell befizetniük.
 - Egyszerre több havi osztálypénzt is be lehet fizetni, ebben az esetben a befizetés 1000 Ft valamely többszöröse. Ez az összes befizetés kb. 20%-ában fordul elő.
 - Lehet túlfizetés is, azaz a tanulók fizethetnek 10 000 Ft-nál többet is egy tanévben.
 - A befizetés a hónap során bármikor történhet, de egy hónapban csak egyszer.
 - Késő tanuló nincs, azaz mindenki minden hónapban befizet legalább 1000 Ft-ot.
- Közös költségek
 - Az osztály programokra, eszközökre havi 2000 és 5000 Ft közötti összeget költ, ennek az egy főre eső részét minden tanuló egyenlegéből le kell vonni.
- Egyedi költségek
 - A tanulónak a tanév során 1-3 alkalommal 1000 és 1500 Ft közötti egyéni kiadásuk is van, amelyet az érintett tanuló egyenlegéből le kell vonni.
- Induláskor a program generálja véletlenszerűen a tanulók egyedi számlaegyenlegét!
- Ezután az *előbb leírtaknak megfelelően* generálja véletlenszerűen és tárolja el a tanévre vonatkozó befizetések, illetve költségek fontos adatait: tanuló sorszáma, dátum, jelleg (befizetés, közös költség, egyéni költség), összeg.
- A program készítsen kimutatást a tanulók éves befizetéseiről és költségeiről, az alábbi minta szerint! A lista a tanulók sorszáma szerint, azon belül a tranzakciók dátuma szerint növekvően legyen rendezve!

Minta:

Tanuló sorszáma: 17/30.

Nyitóegyenleg: 2000 Ft

Befizetések/költségek:

2009.09.01.	Osztálypénz	6000 Ft
2009.09.12.	Közös költség	-100 Ft
2009.09.23.	Egyéni költség	-1000 Ft
2009.10.17.	Közös költség	-200 Ft

...

Záróegyenleg: 6700 Ft

- Megjegyzések:
 - A fenti lista nem teljes, csak a formai megjelenítést illusztrálja!
 - A tanuló sorszáma mellett tüntesse fel az osztály létszámát!

A feladat megoldásaként teljes, fordítható és futtatható kódot kérünk, mely az adatokat a képernyőre (standard output) írja ki. Vizuális fejlesztőeszköz használata esetén az algoritmust konzol alkalmazásként (szöveges ablakban futó) kérjük elkészíteni! Beadandó: a feladatot megoldó program forráskódja.

Mintamegoldás: a feladat egy lehetséges megoldása C# nyelven

(A tördelési problémák miatt a fájlban mellékelt megoldáshoz képest a nagyon hosszú sorok néhány helyen áttördelve láthatók!)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Feladat3
{
    enum jelleg
    { osztalypenz = 0, kozos = 1, egyeni = 2 }

    class Osztalykassza
    {
        private const int maxTranzakciodb = 23;
        private int tanulodb;
        private struct tranzakcio // Egy tranzakcio adatai
        {
            public DateTime datum;
            public jelleg jelleg;
            public int osszeg; }
        private struct tanulo // Egy tanuló adatai
        {
            public int nyitoegyenleg;
            public int zaroegyenleg;
            public int tranzakciodb;
            public tranzakcio[] tranzakciok; }
        private tanulo[] tanulok;

        public void general()
        {
            Random r = new Random();
            tanulodb = r.Next(25,36); // A tanulók száma
            tanulok = new tanulo[tanulodb];
            for (int i = 0; i < tanulodb; i++)
            {
                tanulok[i].nyitoegyenleg = r.Next(1000, 3501);
                tanulok[i].tranzakciok=new tranzakcio[maxTranzakciodb];
                for (int j = 0; j < 10; j++)
                {
                    //Osztálypénz
                    int honap=(j < 4 ? j + 9 : j - 3);
                    tanulok[i].tranzakciok[j].datum =
                        new DateTime((j<4?2009:2010),honap,
                            r.Next(1, DateTime.DaysInMonth(2009, honap)+1));
                    tanulok[i].tranzakciok[j].jelleg = jelleg.osztalypenz;
                    tanulok[i].tranzakciok[j].osszeg =
                        (r.NextDouble() > 0.2 ? 1000: 1000* r.Next(2, 5));
                    // Közös költség
                    if (i == 0)
                    {
                        tanulok[i].tranzakciok[j + 10].datum =
                            new DateTime((j < 4 ? 2009 : 2010), honap,
                                r.Next(1, DateTime.DaysInMonth(2009, honap) + 1));
                        tanulok[i].tranzakciok[j + 10].osszeg =
                            -r.Next(2000, 5001) / tanulodb;
                    }
                    else
                    {
                        tanulok[i].tranzakciok[j + 10].datum =
                            tanulok[i - 1].tranzakciok[j + 10].datum;
                        tanulok[i].tranzakciok[j + 10].osszeg =
                            tanulok[i - 1].tranzakciok[j + 10].osszeg; } // else
                    tanulok[i].tranzakciok[j + 10].jelleg = jelleg.kozos; } // for
                }
            }
        }
    }
}
```

```

        int egyedidb = r.Next(1, 4);
        for (int j = 0; j < egyedidb; j++)
        {
            int ev=(r.NextDouble() < 0.4 ? 2009: 2010);
            int honap=(ev==2009?r.Next(9,13):r.Next(1,7));
            tanulok[i].tranzakciok[j + 20].datum =
                new DateTime(ev,honap,r.Next(1,28));
            tanulok[i].tranzakciok[j + 20].jelleg = jelleg.egyeni;
            tanulok[i].tranzakciok[j + 20].osszeg = -r.Next(1000, 1500); }
        tanulok[i].tranzakciodb = 20 + egyedidb;
        tanulok[i].zaroegyenleg = tanulok[i].nyitoegyenleg;
        for (int j = 0; j < tanulok[i].tranzakciodb; j++)
        { tanulok[i].zaroegyenleg += tanulok[i].tranzakciok[j].osszeg; }
    }
}

public void rendez()
{
    for (int k = 0; k < tanulodb; k++)
    {
        for (int i = 0; i < tanulok[k].tranzakciodb-1; i++)
        {
            for (int j = i + 1; j < tanulok[k].tranzakciodb; j++)
            {
                if (tanulok[k].tranzakciok[i].datum >
                    tanulok[k].tranzakciok[j].datum)
                {
                    tranzakcio s = tanulok[k].tranzakciok[i];
                    tanulok[k].tranzakciok[i] = tanulok[k].tranzakciok[j];
                    tanulok[k].tranzakciok[j] = s;
                }
            }
        }
    }
}

public void kiirat()
{
    for (int i = 0; i < tanulodb; i++)
    {
        Console.WriteLine("\nTanuló sorszáma: " + (i + 1) + "/" + tanulodb);
        Console.WriteLine("    Nyitóegyenleg: " +
            tanulok[i].nyitoegyenleg + " Ft");
        Console.WriteLine("    Befizetések/költségek:");
        for (int j = 0; j < tanulok[i].tranzakciodb; j++)
        { Console.WriteLine(String.Format(
            "{0,10}{1:d}{2,5}{3,-20}{4,15}{5,3}",
            "", tanulok[i].tranzakciok[j].datum, "",
            (tanulok[i].tranzakciok[j].jelleg == jelleg.osztalypenz ?
            "Osztálypénz" :
            (tanulok[i].tranzakciok[j].jelleg == jelleg.kozos ?
            "Közös költség" : "Egyéni költség"))
            , tanulok[i].tranzakciok[j].osszeg, "Ft")); }
        Console.WriteLine("    Záróegyenleg: " +
            tanulok[i].zaroegyenleg + " Ft");
        Console.ReadLine();
    }
}
}

class Program
{
    static void Main(string[] args)
    {
        Osztalykassza o = new Osztalykassza();
        o.general();
        o.rendez();
        o.kiirat();
    }
}

```

Értékelés:

- a) A programkód szintaktikailag hibátlan, lefordítható 1 pont
- Ez a pont csak abban az esetben adható meg, ha a programkód tartalmaz a b-e szakaszokba tartozó, összességében legalább 5 pontot érő részmegoldást.
- b) Adatgenerálás, adatok eltárolása 7 pont
- A tanulók létszámának generálása és tárolása: 1 pont
- Megjegyzés: a pont most és a továbbiakban is csak akkor jár, ha a generálás a feladatleírásban megadott intervallumban történik.*
- Nyitóegyenlegek generálása és tárolása minden tanuló esetében: 1 pont
 - Befizetések generálása és tárolása minden tanuló esetén, minden hónapban pontosan egyszer, dátummal együtt: 1 pont
 - A generált dátumok a program minden részében az iskolai tanévnek megfelelőek: 1 pont
 - A program az esetek kb. 20%-ában többhavi befizetést generál: 1 pont
 - Közös költségek generálása és tárolása minden tanuló esetén, jól kiszámolt negatív értékek, minden hónapban pontosan egyszer, dátummal együtt, minden tanulónak ugyanakkor és ugyanaz az érték: 1 pont
 - Egyéni költségek generálása és tárolása minden tanuló esetén, negatív értékek, a tanév során 1-3 alkalommal, dátummal együtt: 1 pont
- c) Záróegyenlegek kiszámítása..... 1 pont
- A nyitóegyenleg, a befizetések és költségek összegzése minden tanuló esetén
- d) Rendezés..... 2 pont
- A tranzakciók dátum szerinti rendezése legalább egy tanuló esetében: 1 pont
 - A tranzakciók dátum szerinti rendezése minden tanuló esetében: 1 pont
- Megjegyzés: a pont akkor is jár, ha a generálás úgy történik, hogy az adatok a megfelelő rendezettség szerint jönnek létre.*
- e) Táblázatszerű kiírás:..... 4 pont
- A tanulók sorszám szerint növekvő rendezettségben jelennek meg, a sorszám a minta szerinti: 1 pont
 - Látható minden tanulónál a nyitó- és záróegyenleg: 1 pont
 - Minden tanuló esetében megjelenik a dátum, a tranzakció jellege és az összeg: 1 pont
 - A táblázat áttekinthető, a minta szerinti: 1 pont

4. feladat**15 pont****Feladatkitűzés:**

Adott a **mesterek** nevű adatbázis, amely mesteremberek megrendeléseivel kapcsolatos adatokat tartalmaz. Az adatbázis a vizsgabizottság által megadott helyen, MS-Access 2000 formátumú állományban található.

Azok számára, akik az Access formátumát nem ismerő rendszert használnak, az adatbázis tábláit UTF-8 kódolású szöveges állományokban is megadtuk. Ezek első sorában az adott tábla mezőnevei, a többi sorban az adatrekordok találhatók. Az adatokat a sorokon belül pontosvessző határolja el egymástól.

Az adatbázis első sorban feladatkitűzési céllal készült, így nem modellezi tökéletesen a való életben felmerülő összes lehetséges helyzetet.

Az adatbázis az alábbi táblákat és relációkat tartalmazza:

mester (

<u>mesterazon</u>	: Egész szám	-> mester_tevekenyseg.mesterazon
nev	: Szöveg	
varos	: Szöveg	
telefon	: Szöveg	

)**tevekenyseg (**

<u>tevekenysegazon</u>	: Egész szám	-> mester_tevekenyseg.tevekenysegazon
tevekenyseg	: Szöveg	

)**mester_tevekenyseg (**

<u>mtazon</u>	: Egész szám	-> munka.mtazon
mesterazon	: Egész szám	-> mester.mesterazon
tevekenysegazon	: Egész szám	-> tevekenyseg.tevekenysegazon

)**megrendelo (**

<u>megrendeloazon</u>	: Egész szám	-> munka.megrendeloazon
nev	: Szöveg	
varos	: Szöveg	
lakcim	: Szöveg	
telefon	: Szöveg	

)**munka (**

<u>munkaazon</u>	: Egész szám	
megrendeloazon	: Egész szám	-> megrendelo.megrendeloazon
mtazon	: Egész szám	-> mester_tevekenyseg.mtazon
megrenddatum	: Dátum	
teljdatum	: Dátum	
munkadij	: Pénznem	
garancia	: Egész szám	

)

A kettőspont után az adatmező típusát adtuk meg, a „->” karakterek után pedig a más táblákkal való kapcsolatot.

Az elsődleges kulcsot aláhúzás jelöli.



A **mester** adattábla tartalmazza a mesteremberek egyedi azonosítóját, nevét, városát és telefonszámát. A táblában szerepelhet több, ugyanolyan nevű mester, akár azonos városban is.

A **tevekenyse** adattábla tartalmazza a mesterek által végzett tevékenységek azonosítóját és megnevezését.

A **mester_tevekenyse** adattábla (kapcsolótábla) tartalmazza, hogy melyik mester milyen tevékenységet végez. Ugyanaz a mester több különböző tevékenységet is végezhet, és ugyanazt a tevékenységet több különböző mester is végezheti. Minden mester-tevékenység párosítás egyedi azonosítóval rendelkezik.

A **megrendelo** adattábla a megrendelők azonosítóját, nevét, városát, lakcímét és telefonszámát tartalmazza.

A **munka** adattábla tartalmazza az elvégzett munkákkal kapcsolatos adatokat: a megrendelő azonosítóját, a mester-tevékenység azonosítót, a megrendelés és teljesítés dátumát, a munkadíjat és a garancia években mért idejét.

- Készítsen lekérdezést, amely megadja azoknak a mestereknek az azonosítóját, nevét és városát, akik valamelyik munkájukat a megrendeléstől számított 5 napon belül teljesítették, és ugyanarra a munkára 3 évnél kevesebb garanciát vállaltak!
- Siófokon a következő rendelkezést hozták: minden olyan mester, aki legalább 3 különböző tevékenységet végez, az elvégzett munkák összértéke után 10%-os támogatást kap. Készítsen lekérdezést, amely megadja a siófoki mestereknek kifizetett összes támogatás értékét!
- A pécsi mesterek az egyszerűbb kommunikáció érdekében ugyanahhoz a mobil-szolgáltatóhoz szerződnek, de a régi számukat szeretnék megtartani. Készítsen lekérdezést, amely a pécsi mesterek (20) és (30) kezdetű telefonszámaiban a szolgáltató-azonosítót egységesen (70)-re változtatja, a telefonszám további részét pedig változatlanul hagyja! A (70)-es számokat a lekérdezés semmilyen módon ne módosítsa!
 - A telefonszámok 12 mind karakter hosszúak, pl. „(30) 1234567”.
 - A feladat megoldása előtt készítsen másolatot a mestertábláról mester2 néven, és erre vonatkozóan készítse el a lekérdezést!

Megoldás, értékelés:

a) Lekérdezés A 4 pont

- A lista a megadott mezőket tartalmazza¹: 1 pont
- Mindegyik mester csak egyszer jelenik meg a listában²: 1 pont
- A munkavégzés idejére és a garanciára vonatkozó szűrőfeltétel helyes³: 1 pont
- A táblák közötti kapcsolat helyes⁴: 1 pont

Egy lehetséges megoldás MS Access SQL-ben:

Segédlekérdezés (ASeged):

```
SELECT mester.mesterazon, mester.nev, mester.varos1
FROM munka, mester, mester_tevekenyseg
WHERE (mester_tevekenyseg.mtazon=munka.mtazon)4 AND
(mester_tevekenyseg.mesterazon=mester.mesterazon)4 AND
(munka.teljdatum-munka.megrenddatum<=5) AND (garancia<3)3;
```

Fő lekérdezés (A):

```
SELECT mesterazon, nev, varos1
FROM Feladat1a
GROUP BY mesterazon, nev, varos2;
```

b) Lekérdezés B 7 pont

- A városra vonatkozó szűrőfeltétel helyes¹: 1 pont
- A tevékenységek számára vonatkozó szűrőfeltétel helyes²: 1 pont
- A lekérdezés előállítja a feltételnek megfelelő mesterek azonosítóit³: 1 pont
- Csoportosítás a mesterek azonosítója szerint⁴: 1 pont
- A táblák közötti kapcsolat mindenütt helyes⁵: 1 pont
- Szűrés a kiválogatott mesterekre vonatkozóan, a munkadíjak előállításá⁶: 1 pont
- Helyes a számított mező⁷: 1 pont

Egy lehetséges megoldás:

Segédlekérdezés (BSeged):

```
SELECT munka.munkadij6
FROM munka, mester_tevekenyseg
WHERE (mester_tevekenyseg.mtazon=munka.mtazon)5 AND
mester_tevekenyseg.mesterazon In6
(
SELECT mester_tevekenyseg.mesterazon3
FROM mester_tevekenyseg, mester WHERE
(mester.mesterazon=mester_tevekenyseg.mesterazon)5 AND
(mester.varos="Siófok")1
GROUP BY mester_tevekenyseg.mesterazon4
HAVING Count(mester_tevekenyseg.tevekenysegazon)>=32
);
```

Fő lekérdezés (B):

```
SELECT Sum(munkadij)*0.17 AS Tamogatas
FROM BSeged;
```

c) Lekérdezés C 4 pont

- A mester2 tábla létrehozása, frissítő lekérdezés alkalmazása¹: 1 pont
- Helyes a telefonszámra vonatkozó szűrőfeltétel²: 1 pont
- Helyes a városra vonatkozó szűrőfeltétel³: 1 pont
- Helyes a módosítás⁴: 1 pont

```
UPDATE1 mester21 SET telefon = "(70"+Right(mester2.telefon,9)4
WHERE ( Left(mester2.telefon,3)="(30" Or
Left(mester2.telefon,3)="(20" )2 AND mester2.varos="Pécs"3;
```