KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

OBJEKTINIO PROGRAMAVIMO ĮVADAS (P175B157)

Laboratorinių darbų ataskaita

Atliko:

MGDMI-4/2 gr. studentas

Mykolas Logminas

2024 m. lapkričio 26 d.

Priėmė:

Lekt. Tatjana Dulinskienė

KAUNAS 2024

TURINYS

1. Pažintis su klase

1.1. Darbo užduotis

Užduotis(U2-2)

• Sukurkite klasę Studentas, kuri turėtų kintamuosius amžiui ir ūgiui saugoti. Trys studentai nutarė treniruotis žaisti krepšinį. Raskite, koks aukščiausio studento amžius ir koks jauniausio studento ūgis.

• Papildykite klasę Studentas kintamuoju, skirtu studento svoriui saugoti. Sukurkite klasę Liftas, kuri turėtų kintamuosius lifto keliamosios galios reikšmei ir talpai saugoti. Per kelis kartus visi studentai pakils liftu į reikiamą aukštą?

• Papildykite klasę Liftas metodais Dėti(), kurie leistų keisti lifto keliamąją galią ir talpą. Ar visi studentai vienu metu bus pakelti į reikiamą aukštį, jeigu lifto keliamoji galia bus padvigubinta? O jeigu talpa bus padvigubinta?

1.2. Programos tekstas

Program.cs

```
using System;
namespace Laboratorinis
{
    /// <summary>
    /// Klasė studento duomenims aprašyti
    /// </summary> class
    Studentas
    {
        private double studentoUgis; double
        studentoAmzius; double
        studentoSvoris;
        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="studentoUgis">Studento ūgis</param>
        /// <param name="studentoAmzius">Studento amžius</param>
        /// <param name="studentoSvoris">Studento svoris</param>
        public Studentas(double studentoUgis, double studentoAmzius, double studentoSvoris)
        {
            this.studentoUgis = studentoUgis; this.studentoAmzius =
            studentoAmzius; this.studentoSvoris = studentoSvoris;
        }
        /// <summary>
        /// Gražina studento ūgį
        /// </summary>
        /// <returns>Studento ūgis</returns>
        public double Ugis() { return studentoUgis; }
        /// <summary>
        /// Gražina studento amžių
        /// </summary>
        /// <returns>Studento amžius</returns>
        public double Amzius() { return studentoAmzius; }
        /// <summary>
        /// Gražina studento svorį
        /// </summary>
        /// <returns>Studento svoris</returns>
        public double Svoris() { return studentoSvoris; }
    }
    /// <summary>
    /// Klasė liftas duomenims aprašyti
    /// </summary> class
    Liftas
    {
        private double liftoTalpa;
```

```
        double liftoKeliamojiGalia;
        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="liftoTalpa">Lifto talpa</param>
       /// <param name="liftoKeliamojiGalia">Lifto keliamoji galia(kiek liftas gali    pakelti
      kilogramų)</param>
        public Liftas(double liftoTalpa, double liftoKeliamojiGalia)
        {
            this.liftoTalpa = liftoTalpa; this.liftoKeliamojiGalia =
            liftoKeliamojiGalia;
        }
        /// <summary>
        /// Gražina lifto talpą
        /// </summary>
        /// <returns> Lifto talpa</returns>
        public double Talpa() { return liftoTalpa; }
        /// <summary>
        /// Gražina lifto keliamąją galią
        /// </summary>
        /// <returns>Lifto galia</returns>
        public double Galia() { return liftoKeliamojiGalia; }
        /// <summary>
        /// Gražina lifto talpą(duoti)
        /// </summary>
        /// <returns>Lifto talpą *2</returns>
        public double DuotiTalpa() { return liftoTalpa * 2; }
        /// <summary>
        /// Gražina lifto galią(duoti)
        /// </summary>
        /// <returns>Lifto talpą *2</returns>
        public double DuotiGalia() { return liftoKeliamojiGalia * 2; }
}
/// <summary>
/// Klasė užduotyje nurodytiems skaičiavimams atlikti
/// </summary> internal class
Program
{
        static void Main(string[] args)
        {
            Studentas studentas1, studentas2, studentas3;
            studentas1 = new Studentas(180, 20, 75);//Pirmo studento parametrai
            studentas2 = new Studentas(190, 19, 80);//Antro studento parametrai
            studentas3 = new Studentas(175, 18, 70);//Trečio studento parametrai Liftas liftas;
            liftas = new Liftas(5, 400);//Lifto parametrai
            double maxUgis = 0, minAmzius = 0, bendrasSvoris = 0;//maxUgis Aukščiausio studento ūgis,
            minAmzius - Jauniausio studento amžius, bendrasSvoris -Studentų svorių suma
           int asmenys = 3, kilimuSkaicius = 0, kilimuSkaicius2 = 0;//asmenys -Asmenų skaičius,
          kilimuSkaicius -liftu kilimų skaičius, kilimuSkaicius2 -liftukilimų skaičius
            Console.WriteLine("{0,21}", "Duomenys"); Console.WriteLine("              ");
           Console.WriteLine("  Studentas {0,6}  {1,2}  {2,0}",
          "Ūgis", "Amžius", "Svoris");
            Console.WriteLine("        ----------------------------------------------------");
            Console.WriteLine(" Studentas 1  {0}  {1,4}  {2,6}  ", studentas1.Ugis(),
          studentas1.Amzius(), studentas1.Svoris()); Console.WriteLine(" Studentas 2  {0}  {1,4}
          {2,6}  ", studentas2.Ugis(), studentas2.Amzius(), studentas2.Svoris());
          Console.WriteLine(" Studentas 3  {0}  {1,4}  {2,6}  ", studentas3.Ugis(),
          studentas3.Amzius(), studentas3.Svoris()); Console.WriteLine("              ");
            Console.WriteLine("");  ----------------------------------------------------
```

```
Console.WriteLine("{0,14} {1,7}", "Talpa", "Galia"); Console.WriteLine("
                    ------------------------------------------------");
Console.WriteLine(" Liftas {0,2} {1,9}", liftas.Talpa(), liftas.Galia());
Console.WriteLine("       ------------------------------------------------");
Console.WriteLine(""); Console.WriteLine("{0,35}",
"Rezultatai"); Console.WriteLine("
---------------"); ------------------------------------------------------------------------
if (liftas.Galia() > 0 && liftas.Galia() != 0 && liftas.Talpa() > 0 && liftas.Talpa() != 0 &&
    studentas1.Ugis() > 0 && studentas1.Ugis() != 0 &&
    studentas2.Ugis() > 0 && studentas2.Ugis() != 0 && studentas3.Ugis() > 0 &&
     studentas3.Ugis() != 0 && studentas1.Amzius() > 0 && studentas3.Ugis()
     != 0 && studentas2.Amzius() > 0 && studentas2.Amzius() != 0 && studentas3.Amzius() > 0
    &&studentas3.Amzius() != 0 && studentas1.Svoris() > 0 && studentas1.Svoris() != 0 &&
    studentas2.Svoris() > 0 && studentas2.Svoris() != 0 && studentas3.Svoris() > 0 &&
    studentas3.Svoris() != 0)//tirktina ar parametrai yra teigiami ir nėra lygūs 0
{
      Console.WriteLine("Aukščiausio studento
                  amžius:{0,30}",auksciausioStudentoAmzius(studentas1,studentas2,
                   studentas3,maxUgis, minAmzius));
      Console.WriteLine("Jauniausio studento ūgis: {0,33}", jauniausioStudentoUgis(studentas1,
                  studentas2, studentas3, maxUgis, minAmzius));
      Console.WriteLine("Studentai liftu užkils per: {0,24} kartus",
                  liftuKilimai(studentas1, studentas2, studentas3,liftas, bendrasSvoris,
                  asmenys, kilimuSkaicius,kilimuSkaicius2));
      //---Tikrina ar pakeitus lifto keliamąją galią ir talpą studentai     pakiltų liftu per
      vieną kartą---
      if (liftas.DuotiTalpa() >= asmenys && liftas.Galia() >= bendrasSvoris)
      {
            Console.WriteLine("Pakeitus lifto talpą, visi studentai pakils per vieną kartą");
      }
      if (liftas.DuotiTalpa() < asmenys && liftas.Galia() >= bendrasSvoris)
      {
            Console.WriteLine("Pakeitus lifto talpą, visi studentai nepakils per vieną kartą");
      }
      if (liftas.Talpa() >= asmenys && liftas.DuotiGalia() >= bendrasSvoris)
      {
            Console.WriteLine("Pakeitus lifto galią, visi studentai pakils per vieną kartą");
      }
      if (liftas.DuotiTalpa() >= asmenys && liftas.Galia() < bendrasSvoris)
      {
            Console.WriteLine("Pakeitus lifto galią, visi studentai nepakils per vieną
                  kartą");
      }

}
```

```
        else Console.WriteLine("KLAIDA - parametras negali būti lygus 0 ar neigiamas
                                skaičius");//Išvedimas, jei bent vienas parametras yra neigiams ar lygus


    }
    /// <summary>
    ///Randa aukščiausio studento amžių
    /// </summary>
    /// <param name="studentas1">Pirmo studento parametrai</param>
    /// <param name="studentas2">Antro studento parametrai</param>
    /// <param name="studentas3">Trečio studento parametrai</param>
    /// <param name="maxUgis">Didžiausio studento ūgis</param>
    /// <param name="minAmzius">Jauniausio studento amžius</param>
    /// <returns>Gražina aukščiausio studento amžių</returns>
    static double auksciausioStudentoAmzius(Studentas studentas1, Studentas studentas2, Studentas
studentas3, double maxUgis, double minAmzius)
    {
        maxUgis = Math.Max(studentas1.Ugis(), Math.Max(studentas2.Ugis(),
                                studentas3.Ugis()));
        if (studentas1.Ugis() == maxUgis)
        {
            return studentas1.Amzius();
        }
        else
        if (studentas2.Ugis() == maxUgis)
        {
            return studentas2.Amzius();
        }
        else return studentas3.Amzius();
    }
    /// <summary>
    /// Randa jauniausio studento ūgį
    /// </summary>
    /// <param name="studentas1">Pirmo studento parametrai</param>
    /// <param name="studentas2">Antro studento parametrai</param>
    /// <param name="studentas3">Trečio studento parametrai</param>
    /// <param name="maxUgis">Didžiausio studento ūgis</param>
    /// <param name="minAmzius">Jauniausio studento amžius</param>
    /// <returns>Gražina jauniausio studento ūgį</returns>
    static double jauniausioStudentoUgis(Studentas studentas1, Studentas studentas2, Studentas
studentas3, double maxUgis, double minAmzius)
    {
        minAmzius = Math.Min(studentas1.Amzius(), Math.Min(studentas2.Amzius(),
       studentas3.Amzius()));
        if (studentas1.Amzius() == minAmzius)
        {
            return studentas1.Ugis();
        }
        else
        if (studentas2.Amzius() == minAmzius)
        {
            return studentas2.Ugis();
        }
        else return studentas3.Ugis();
    }
    /// <summary>
    /// Suskaičiuoja per kiek kilimų studentai pakils į reikiamą aukštą
    /// </summary>
    /// <param name="studentas1">Pirmo studento parametrai</param>
    /// <param name="studentas2">Antro studento parametrai</param>
    /// <param name="studentas3">Trečio studento parametrai</param>
    /// <param name="liftas">Lifto parametrai</param>
    /// <param name="bendrasSvoris">Studentų svorių suma</param>
```

```
        /// <param name="asmenys">Kiek asmenų kyla liftu</param>
        /// <param name="kilimuSkaicius">Kilimų liftu skaičius</param>
        /// <param name="kilimuSkaicius2">Kilimų liftu skaičius</param>
        /// <returns>Gražina lifto kilimų skaičių</returns>
        static double liftuKilimai(Studentas studentas1, Studentas studentas2,
                Studentas studentas3, Liftas liftas, double bendrasSvoris, int asmenys = 3,     int
        kilimuSkaicius = 0, int kilimuSkaicius2 = 0)
        {
            bendrasSvoris = studentas1.Svoris() + studentas2.Svoris() + studentas3.Svoris();
            if (liftas.Talpa() >= asmenys && liftas.Galia() >= bendrasSvoris)
            {
                return 1;
            }
            else if (liftas.Talpa() < asmenys)
            {
                while (liftas.Talpa() < asmenys)
                {
                    asmenys = (int)(asmenys - liftas.Talpa()); kilimuSkaicius++;
                }
                kilimuSkaicius++; return
                kilimuSkaicius;
            }
            else if (liftas.Galia() < bendrasSvoris)
            {

                while (liftas.Galia() < bendrasSvoris)
                {
                    bendrasSvoris = bendrasSvoris - liftas.Galia(); kilimuSkaicius++;
                }
                kilimuSkaicius++; return
                kilimuSkaicius;
            }
            else
            {
                while (liftas.Talpa() < asmenys)
                {
                    asmenys = (int)(asmenys - liftas.Talpa()); kilimuSkaicius++;
                }
                kilimuSkaicius++;
                while (liftas.Galia() < bendrasSvoris)
                {
                    bendrasSvoris = bendrasSvoris - liftas.Galia(); kilimuSkaicius2++;
                }
                kilimuSkaicius2++;
                return Math.Max(kilimuSkaicius, kilimuSkaicius2);
            }
        }
    }
}
```

1.3. Pradiniai duomenys ir rezultatai

```
               Duomenys
-------------------------------------------------
  Studentas    Ugis    Amzius  Svoris
-------------------------------------------------
 Studentas 1  180     20       75
 Studentas 2  190     19       80
 Studentas 3  175     18       70
-------------------------------------------------

          Talpa    Galia
-------------------------------------------------
 Liftas  5        400
-------------------------------------------------

                           Rezultatai
-------------------------------------------------------------------------------------
Auksciausio studento amzius:                                          19
Jauniausio studento ugis:                                            175
Studentai liftu uzkils per:                                    1 kartus
Pakeitus lifto talpa, visi studentai pakils per viena karta
Pakeitus lifto galia, visi studentai pakils per viena karta
-------------------------------------------------------------------------------------
```

```
               Duomenys
-------------------------------------------------
  Studentas    Ugis    Amzius  Svoris
-------------------------------------------------
 Studentas 1  180     20       75
 Studentas 2  190      0       80
 Studentas 3  175     18       70
-------------------------------------------------

          Talpa    Galia
-------------------------------------------------
 Liftas  5        400
-------------------------------------------------

                           Rezultatai
-------------------------------------------------------------------------------------
KLAIDA - parametras negali buti lygus 0 ar neigiamas skaicius
-------------------------------------------------------------------------------------
```

1.4. Dėstytojo pastabos

2. Objektų rinkinys

2.1. Darbo užduotis

Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Sukurkite klasę Krepšininkas, kuri turėtų kintamuosius vardui su pavarde, amžiui bei ūgiui saugoti. Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis.

Papildykite programą veiksmais su dviejų krepšinio mokyklų duomenimis. Kiekvienos mokyklos duomenys saugomi atskiruose failuose. Kurioje mokykloje aukščiausias sportininkas? Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.

2.2. Programos tekstas

```
using System.IO;
namespace P3_labaratorinis
{
    /// <summary>
    /// Krepšininkų klasė
    /// </summary> class
    Krepsininkai
    {
        private string VardasPavarde; private int
        Amzius;
        private double Ugis;
        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="vardasPavarde">Krepšininko vardas ir pavardė</param>
        /// <param name="amzius">Krepšininko amžius</param>
        /// <param name="ugis">Krepšininko ūgis</param>
        public Krepsininkai(string vardasPavarde, int amzius, double ugis)
        {
            this.VardasPavarde = vardasPavarde; this.Amzius
            = amzius;
            this.Ugis = ugis;
        }
        /// <summary>
        /// Gražina vardą pavardę
        /// </summary>
        /// <returns>Vardas pavardė</returns>
        public string vardasPavarde() { return VardasPavarde; }
        /// <summary>
        /// Gražina amžių
        /// </summary>
        /// <returns>Amžius</returns>
        public int amzius() { return Amzius; }
        /// <summary>
        /// Gražina ūgį
        /// </summary>
        /// <returns>Ūgis</returns
        public double ugis() { return Ugis; }
    }
    internal class Program
    {
        const string Df1 = "duom1.txt"; const string Df2
        = "duom2.txt";
        const string Rez = "..\\..\\rez.txt"; const int Cn = 1000;
        static void Main(string[] args)
        {
```

```
        string pavadinimas1, pavadinimas2;
        Krepsininkai[] krepsininkai1 = new Krepsininkai[Cn]; Krepsininkai[]
        krepsininkai2 = new Krepsininkai[Cn]; Krepsininkai[] bendras = new
        Krepsininkai[Cn]; Krepsininkai[] auksciausi = new Krepsininkai[Cn];
        int kiekAuksciausi, kiekBendras = 0;
        if (File.Exists(Rez))
            File.Delete(Rez);
        Nuskaitymas(out kiek1, krepsininkai1, Df1, out pavadinimas1); DuomenuIsvedimas(kiek1,
        krepsininkai1, Rez, pavadinimas1); Nuskaitymas(out kiek2, krepsininkai2, Df2, out
        pavadinimas2); DuomenuIsvedimas(kiek2, krepsininkai2, Rez, pavadinimas2);
        BendrasSarasas(ref kiekBendras, bendras, krepsininkai1, kiek1); BendrasSarasas(ref
        kiekBendras, bendras, krepsininkai2, kiek2); AuksciausiuSarasas(kiekBendras, out
        kiekAuksciausi, bendras,
            auksciausi);
        VidurkiuIsvedimas(kiek1,      krepsininkai1,      Rez,      pavadinimas1);
        VidurkiuIsvedimas(kiek2,      krepsininkai2,      Rez,      pavadinimas2);
        AukstuIsvedimas(kiek1, kiek2, krepsininkai1, krepsininkai2,
            pavadinimas1, pavadinimas2, Rez, auksciausi, kiekAuksciausi);
}
/// <summary>
/// Nuskaito pradinius duomenis
/// </summary>
/// <param name="kiek">krepšinininkų kiekis</param>
/// <param name="krepsininkai">krepšininkų masyvas</param>
/// <param name="Df">duomenų failas</param>
/// <param name="pavadinimas">komados pavadinimas</param>
private static void Nuskaitymas(out int kiek, Krepsininkai[] krepsininkai, string Df, out string
        pavadinimas)
{
        kiek = 0; string[]
        dalys;
        string vardasPavarde; int
        amzius = 0; double ugis = 0;
        using (StreamReader skaitymas = new StreamReader(Df))
        {
            string laikinas = skaitymas.ReadLine(); pavadinimas =
            laikinas;
            laikinas = skaitymas.ReadLine(); while
            (laikinas != null)
            {
                dalys = laikinas.Split(' ');
                vardasPavarde = dalys[0] + " " + dalys[1];
                if (dalys.Length >= 4) // jei yra ir amžius ir ūgis
                {
                    amzius = int.Parse(dalys[2]); ugis =
                    double.Parse(dalys[3]);
                }
                krepsininkai[kiek] = new Krepsininkai(vardasPavarde, amzius, ugis);
                kiek++;
                laikinas = skaitymas.ReadLine();
            }
        }
}
/// <summary>
/// Skaičiuoja amžiaus vidukį
/// </summary>
/// <param name="krepsininkai">krepšininkų masyvas</param>
/// <param name="kiek">krepšininkų kiekis</param>
/// <returns>Amžiaus vidurkis</returns>
static double AmziausVidurkis(Krepsininkai[] krepsininkai, int kiek)
{
        double amziausVid = 0;
```

```
        for (int i = 0; i < kiek; i++)
        {
            amziausVid += krepsininkai[i].amzius();
        }
        amziausVid = amziausVid / kiek; return
        amziausVid;
}
/// <summary>
/// Sudeda krepšininkus į bendrą masyvą
/// </summary>
/// <param name="kiekBendras">kiek bendrų krepšininkų</param>
/// <param name="Bendras">bendrų krepšininkų masyvas</param>
/// <param name="krepsininkai">krepšininkų masyvas(komadų)</param>
/// <param name="kiek">krepšininkų kiekis(komandoj)</param>
private static void BendrasSarasas(ref int kiekBendras, Krepsininkai[] Bendras, Krepsininkai[]
            krepsininkai, int kiek)
{
        for (int i = 0; i < kiek; i++)
        {
            Bendras[kiekBendras] = krepsininkai[i]; kiekBendras++;
        }
}
/// <summary>
/// Suranda aukščiausio krepšininko indeksą
/// </summary>
/// <param name="kiek">krepšininkų kiekis</param>
/// <param name="krepsininkai">krepšininkų masyvas</param>
/// <returns> aukščiusio žaidėjo indeksas</returns>
static int AuksciausiasZaidejas(int kiek, Krepsininkai[] krepsininkai)
{
        int nr = 0;
        for (int i = 0; i < kiek; i++)
        {
            if (krepsininkai[nr].ugis() < krepsininkai[i].ugis())
            {
                nr = i;
            }
        }
        return nr;
}
/// <summary>
/// Suranda aukščiausio krepšininko komandą
/// </summary>
/// <param name="kiek1">kiek krepšininkų pirmoj komandoj</param>
/// <param name="kiek2">kiek krepšininkų antroj komandoj</param>
/// <param name="krepsininkai1">pirmos komados krepšininkų masyvas</param>
/// <param name="krepsininkai2">antros komados krepšininkų masyvas</param>
/// <param name="pavadinimas1">pirmos komados pavadinimas</param>
/// <param name="pavadinimas2">antros komados pavadinimas</param>
/// <returns>aukščiausio krepšininko komanda</returns>
static string AuksciausioZaidejoMokykla(int kiek1, int kiek2, Krepsininkai[]
        krepsininkai1, Krepsininkai[] krepsininkai2, string pavadinimas1, string
        pavadinimas2)
{
        string pavadinimas = "Nėra";
        if (krepsininkai1[AuksciausiasZaidejas(kiek1, krepsininkai1)].ugis()>0
                &&krepsininkai2[AuksciausiasZaidejas(kiek2,krepsininkai2)].ugis()> 0)
        {
            if(krepsininkai1[AuksciausiasZaidejas(kiek1,
                krepsininkai1)].ugis()>krepsininkai2[AuksciausiasZaidejas(kiek2
                ,krepsininkai2)].ugis())
            {
                pavadinimas = pavadinimas1;
            }
```

```
            else if (krepsininkai1[AuksciausiasZaidejas(kiek1, krepsininkai1)].ugis()<
                    krepsininkai2[AuksciausiasZaidejas(kiek2, krepsininkai2)].ugis())
            {
                pavadinimas = pavadinimas2;
            }
            else if (krepsininkai1[AuksciausiasZaidejas(kiek1, krepsininkai1)].ugis() ==
                krepsininkai2[AuksciausiasZaidejas(kiek2, krepsininkai2)].ugis())
            {
                pavadinimas = "Abiejų mokyklų aukščiausi žaidėjai yra vienodo ūgio";
            }
        }
        return pavadinimas;
}
/// <summary>
/// Suskaičiuoja ūgių vidurkį
/// </summary>
/// <param name="bendras">visų krepšininkų masyvas</param>
/// <param name="kiekBendras">kiek iš viso yra krepšininkų</param>
/// <returns> ūgio vidurkis</returns>
static double UgiuVidurkis(Krepsininkai[] bendras, int kiekBendras)
{
        double ugioVid = 0;
        for (int i = 0; i < kiekBendras; i++)
        {
            ugioVid += bendras[i].ugis();
        }
        ugioVid = ugioVid / kiekBendras; return ugioVid;
}
/// <summary>
/// Suranda krepšininkus aukštesnius už ūgio vidurkį
/// </summary>
/// <param name="kiekBendras">kiek iš viso yra krepšininkų</param>
/// <param name="kiekAukstu">kiek krepšininkų aukštesnių už vidurkį</param>
/// <param name="bendras">visų krepšininkų masyvas</param>
/// <param name="auksti">krepšininkų aukštesnių už ūgio vidurkį
        masyvas</param>
private static void AuksciausiuSarasas(int kiekBendras, out int kiekAukstu, Krepsininkai[]
            bendras, Krepsininkai[] auksti)
{
        kiekAukstu = 0;
        for (int i = 0; i < kiekBendras; i++)
        {
            if (bendras[i].ugis() > UgiuVidurkis (bendras, kiekBendras))
            {
                auksti[kiekAukstu] = bendras[i]; kiekAukstu++;
            }
        }
}
/// <summary>
/// Isveda mokyklų amžiaus ir ūgio vidurkius
/// </summary>
/// <param name="kiek">krepšininkų kiekis</param>
/// <param name="krepsininkai">krepšininkų masyvas</param>
/// <param name="Rez">rezultatų failas</param>
/// <param name="pavadinimas">komandos pavadinimas</param> private static
void VidurkiuIsvedimas(int kiek, Krepsininkai[]
            krepsininkai, string Rez, string pavadinimas)
{
```

```
        const string Lentele =
        "| ---------------------------------------------------------------------------------|\n"
        + "| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |\n"
        + "|  ------------------------------| -------------------------| ---------------------|";
        using (var rez = System.IO.File.AppendText(Rez))
        {
            rez.WriteLine(Lentele);
            if (AmziausVidurkis(krepsininkai, kiek) > 0 && UgiuVidurkis
                (krepsininkai, kiek) > 0 && kiek > 0)
            {
                rez.WriteLine("| {0,20} | {1,16:f2} | {2,13:f2} |", pavadinimas,
                    AmziausVidurkis(krepsininkai, kiek), UgiuVidurkis (krepsininkai, kiek));

            }
            else
            {   rez.WriteLine("| {0,20} | {1,16} | {2,13} |", pavadinimas,
                  "Nėra", "Nėra");

            }
            rez.WriteLine("|                  ------------------------------| -------------------------| ------------
               -------                                                    |");
            rez.WriteLine();
        }
}
/// <summary>
/// Spaudina pradinius duomenis
/// </summary>
/// <param name="kiek">krepšininkų kiekis</param>
/// <param name="krepsininkai">krepšininkų masyvas</param>
/// <param name="Rez">rezultatų failas</param>
/// <param name="pavadinimas">komandos pavadinimas</param> private static
void DuomenuIsvedimas(int kiek, Krepsininkai[]
            krepsininkai, string Rez, string pavadinimas)
{
        const string Lentele =
        "| ---------------------------------------------------------|\n"
        + "|         Vardas Pavardė           | Amžius |  Ūgis  |\n"
        + "|  ----------------------------| -----------| -----------|"; using
        (var rez = System.IO.File.AppendText(Rez))
        {
            rez.WriteLine("|          ---------------------------------------------------------|");
            rez.WriteLine("|{0,22} {1,16}", pavadinimas, "|"); rez.WriteLine(Lentele);
            for (int i = 0; i < kiek; i++)
            {
                if (krepsininkai[i].amzius() > 0 && krepsininkai[i].ugis()> 0)
                {
                    rez.WriteLine("| {0,18} {1} {2,6} {3} {4:f2} |",
                        krepsininkai[i].vardasPavarde(),"|", krepsininkai[i].amzius(), "|",
                        krepsininkai[i].ugis());
                }
                else
                {
                    rez.WriteLine("| {0,18} {1} {2,6} {3} {4,5}  |",
                        krepsininkai[i].vardasPavarde(), "|", "Nėra", "|",
                       "Nėra");
                }
            }
            rez.WriteLine("|          ------------------------------| -----------| -----------|");
            rez.WriteLine();
        }
}
/// <summary>
/// Išveda aukščiausio žaidėjo mokyklą ir jo parametrus
/// </summary>
```

```
        /// <param name="kiek1">kiek krepšininkų pirmoj komandoj</param>
        /// <param name="kiek2">kiek krepšininkų antroj komandoj</param>
        /// <param name="krepsininkai1">pirmos komados krepšininkų masyvas</param>
        /// <param name="krepsininkai2">antros komados krepšininkų masyvas</param>
        /// <param name="pavadinimas1">pirmos komados pavadinimas</param>
        /// <param name="pavadinimas2">antros komados pavadinimas</param>
        /// <param name="Rez">rezultatų failas</param>
        /// <param name="auksti">krepšinininkų aukštesnių už ūgio vidurkį
              masyvas</param>
        /// <param name="kiekAukstu">kiek krepšininkų aukštesnių už vidurkį</param>
        private static void AukstuIsvedimas(int kiek1, int kiek2, Krepsininkai[] krepsininkai1, Krepsininkai[]
                krepsininkai2, string pavadinimas1, string pavadinimas2, string Rez, Krepsininkai[] auksti,
                int kiekAukstu)
        {
            const string Lentele =
            "| ------------------------------------------------------|\n"
            + "|            Vardas Pavardė | Amžiaus |  Ūgis  |\n"
            + "| --------------------------| --------| -------| ";
            using (var rez = System.IO.File.AppendText(Rez))
            {
                rez.WriteLine("Aukščiausią žaidėją turinti mokykla: {0}",
                    AuksciausioZaidejoMokykla(kiek1, kiek2, krepsininkai1, krepsininkai2,
                    pavadinimas1, pavadinimas2));
                rez.WriteLine();
                rez.WriteLine("Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas: ");
                rez.WriteLine(Lentele); if
                (kiekAukstu > 0)
                {
                    for (int i = 0; i < kiekAukstu; i++)
                    {
                        if (auksti[i].amzius() > 0 && auksti[i].ugis() > 0)
                        {
                            rez.WriteLine("| {0,18} | {1,7} | {2:f2} |", auksti[i].vardasPavarde(),
                                auksti[i].amzius(), auksti[i].ugis());
                        }
                        else
                        {
                            i = kiekAukstu;
                            rez.WriteLine("| {0,18} | {1,7} | {2,6} |", "Nėra", "Nėra", "Nėra");
                        }

                    }
                }
                else
                {
                    rez.WriteLine("| {0,18} | {1,7} | {2,5}  |", "Nėra", "Nėra", "Nėra");
                }
                rez.WriteLine("|          --------------------------| --------| -------|");
            }
        }
    }
}
```

2.3. *Pradiniai duomenys ir rezultatai*

Įprasti duomenys:
Duomenų failas 1:

```
KTU
Jonas Jonaitis 19 195
```

```
Petras Petraitis 17 200
```

Duomenų failas 2:

```
VU
Antanas Antanaitis 18 200
Tomas Tomaitis 20 205
```

Rezultatų failas:

```
|-----------------------------------------------------|
|                       KTU                           |
|-----------------------------------------------------|
|    Vardas Pavardė     | Amžius |  Ūgis   |
|-----------------------|--------|---------|
|      Jonas Jonaitis   |     19 | 195.00  |
|    Petras Petraitis   |     17 | 200.00  |
|-----------------------|--------|---------|

|-----------------------------------------------------|
|                        VU                           |
|-----------------------------------------------------|
|    Vardas Pavardė     | Amžius |  Ūgis   |
|-----------------------|--------|---------|
| Antanas Antanaitis    |     18 | 200.00  |
|      Tomas Tomaitis   |     20 | 205.00  |
|-----------------------|--------|---------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|----------------------|------------------|---------------|
|                  KTU |               18 |        197.50 |
|----------------------|------------------|---------------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|----------------------|------------------|---------------|
|                   VU |               19 |        202.50 |
|----------------------|------------------|---------------|

Auksčiausią žaidėją turinti mokykla: VU

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:
|-----------------------------------------------|
|      Vardas Pavardė | Amžiaus |  Ūgis   |
|---------------------|---------|---------|
|      Tomas Tomaitis |      20 | 205.00  |
|---------------------|---------|---------|
```

Duomenys lygūs nuliui:

Duomenų failas 1:

```
KTU
Jonas Jonaitis 0 0
Petras Petraitis 0 0
```

Duomenų failas 2:

```
VU
Antanas Antanaitis 0 0
Tomas Tomaitis 0 0
```

Reazultatų failas:

```
|-------------------------------------------------|
|                     KTU                         |
|-------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|---------------------|--------|---------|
|     Jonas Jonaitis  |  Nėra  |  Nėra   |
|   Petras Petraitis  |  Nėra  |  Nėra   |
|---------------------|--------|---------|

|-------------------------------------------------|
|                     VU                          |
|-------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|---------------------|--------|---------|
| Antanas Antanaitis  |  Nėra  |  Nėra   |
|     Tomas Tomaitis  |  Nėra  |  Nėra   |
|---------------------|--------|---------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|----------------------|------------------|-----------------|
|                  KTU |             Nėra |            Nėra |
|----------------------|------------------|-----------------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|----------------------|------------------|-----------------|
|                   VU |             Nėra |            Nėra |
|----------------------|------------------|-----------------|

Auksčiausią žaidėją turinti mokykla: Nėra

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:
|-------------------------------------------|
|      Vardas Pavardė | Amžiaus |  Ūgis   |
|---------------------|---------|---------|
|                Nėra |    Nėra |  Nėra   |
|---------------------|---------|---------|
```

Duomenys neigiami:

Duomenų failas 1:

```
KTU
Jonas Jonaitis -10 -100
Petras Petraitis -10 -100
```

Duomenų failas 2:

```
VU
Antanas Antanaitis -10 -100
Tomas Tomaitis -10 -100
```

Rezultatų failas:

```
|-------------------------------------------------|
|                     KTU                         |
|-------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|---------------------|--------|---------|
|     Jonas Jonaitis  |  Nėra  |  Nėra   |
|   Petras Petraitis  |  Nėra  |  Nėra   |
```

```
|--------------------------|-----------|-----------|

|-------------------------------------------------|
|                      VU                         |
|-------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|--------------------------|-----------|-----------|
| Antanas Antanaitis |    Nėra  |  Nėra  |
|      Tomas Tomaitis |    Nėra |  Nėra  |
|--------------------------|-----------|-----------|

|-------------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|------------------------------|-------------------------|--------------------|
|                     KTU |              Nėra |           Nėra |
|------------------------------|-------------------------|--------------------|

|-------------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|------------------------------|-------------------------|--------------------|
|                      VU |              Nėra |           Nėra |
|------------------------------|-------------------------|--------------------|

Auksčiausią žaidėją turinti mokykla: Nėra

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:
|-----------------------------------------------------|
|      Vardas Pavardė | Amžiaus |  Ūgis  |
|--------------------------|------------|-----------|
|                 Nėra |    Nėra |  Nėra  |
|--------------------------|------------|-----------|
```

Duomenų nėra:

Duomenų failas 1:

```
KTU
Jonas Jonaitis
Petras Petraitis
```

Duomenų failas 2:

```
VU
Antanas Antanaitis
Tomas Tomaitis
```

Reazultatų failas:

```
|-----------------------------------------------------|
|                     KTU                             |
|-----------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|--------------------------|-----------|-----------|
|      Jonas Jonaitis |    Nėra |  Nėra  |
|   Petras Petraitis |    Nėra |  Nėra  |
|--------------------------|-----------|-----------|

|-----------------------------------------------------|
|                      VU                             |
|-----------------------------------------------------|
|   Vardas Pavardė    | Amžius |  Ūgis   |
|--------------------------|-----------|-----------|
| Antanas Antanaitis |     Nėra |  Nėra  |
|      Tomas Tomaitis |    Nėra |  Nėra  |
```

```
|---------------------------|-----------|-----------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|------------------------------|-------------------------|--------------------|
|                     KTU |              Nėra |           Nėra |
|------------------------------|-------------------------|--------------------|

|-------------------------------------------------------------------------|
| Mokyklos pavadinimas | Amžiaus vidurkis | Ūgio vidurkis |
|------------------------------|-------------------------|--------------------|
|                      VU |              Nėra |           Nėra |
|------------------------------|-------------------------|--------------------|

Auksčiausią žaidėją turinti mokykla: Nėra

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:
|-----------------------------------------------------|
|      Vardas Pavardė | Amžiaus |  Ūgis  |
|---------------------------|------------|-----------|
|                 Nėra |    Nėra |  Nėra  |
|---------------------------|------------|-----------|
```

2.4. Dėstytojo pastabos

3. Konteinerinė klasė

3.1. Darbo užduotis

Norėdamas palyginti mobiliojo ryšio operatorių siūlomas išankstinio mokėjimo korteles Sirvydas surinko šią informaciją į tekstinį failą. Faile eilutėmis yra kortelių duomenys: kortelės(tinklo) pavadinimas, pradinė suma kortelėje, tarifas savame tinkle, tarifas į kitus tinklus, SMS žinučių tarifas savame tinkle ir į kitus tinklus. Parašykite programą, kuri spausdintų kortelių duomenis lentele, surastų kortelę, kurios SMS žinučių tarifai į kitus tinklus mažiausi. Papildykite programą veiksmais, kurie leistų atrinkti korteles, kurios leidžia skambinti ir siųsti SMS žinutes savame tinkle nemokamai, ir šį sąrašą surikiuoti pagal pradinę sumą mažėjimo tvarka ir kortelės pavadinimą abėcėliškai.

3.2. Programos tekstas

```
using System;

namespace P4_labaratorinis
{
    /// <summary>
    /// Klase Kortele
    /// </summary> internal class
    Kortele
    {
        private string Pavadinimas;
        private double PradineSum, NamieSkambuciuTarifas, SveturSkambuciuTarifas, NamieSMSTarifas,
                       SveturSMSTarifas;
        /// <summary>
        /// Konstruktorius be parametru, su numatytomis reiksmemis
        /// </summary> public
        Kortele()
        {
            Pavadinimas = "";
            PradineSum = 0;
            NamieSkambuciuTarifas = 0;
            SveturSkambuciuTarifas = 0;
            NamieSMSTarifas = 0;
            SveturSMSTarifas = 0;
        }
        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="Pavadinimas">Kompanijos pavadinimas</param>
        /// <param name="PradineSum">Korteles pradine suma</param>
        /// <param name="NamieSkambuciuTarifas">Skambuciu tarifas skambinant savam
                                                tinkle</param>
        /// <param name="SveturSkambuciuTarifas">Skambuciu tarifas skambinant į
                                                kitus tinklus</param>
        /// <param name="NamieSMSTarifas">Žinučių tarifas rašant savam
                                                tinkle</param>
        /// <param name="SveturSMSTarifas">Žinučių tarifas rašant į kitus
                                                tinklus</param>
        public Kortele(string Pavadinimas, double PradineSum, double NamieSkambuciuTarifas, double
                SveturSkambuciuTarifas, double NamieSMSTarifas, double SveturSMSTarifas)
        {
            this.Pavadinimas = Pavadinimas; this.PradineSum
            = PradineSum;
            this.NamieSkambuciuTarifas = NamieSkambuciuTarifas;
            this.SveturSkambuciuTarifas = SveturSkambuciuTarifas; this.NamieSMSTarifas
            = NamieSMSTarifas; this.SveturSMSTarifas = SveturSMSTarifas;
        }
        /// <summary>
```

```
/// Gražina kompnijos pavadinimą
/// </summary>
/// <returns>Pavadinimas</returns>
public string pavadinimas() { return Pavadinimas; }
/// <summary>
/// Gražina korteles pradine pinigu suma
/// </summary>
///  <returns>pradine  pinigu  suma</returns> public double
pradineSum() { return PradineSum; }
/// <summary>
/// Gražina kortelės skambučių tarifą skambinant savam tinkle
/// </summary>
///  <returns>skambučių  tarifas  skambinant  savam  tinkle</returns> public double
namieSkambuciuTarifas() { return NamieSkambuciuTarifas; }
/// <summary>
/// Gražina kortelės skambučių tarifą skambinant į kitą tinklą
/// </summary>
/// <returns>skambučių tarifas skambinant į kitą tinklą</returns>
public double sveturSkambuciuTarifas() { return SveturSkambuciuTarifas; }
/// <summary>
/// Gražina kortelės žinučių tarifą rašant savam tinkle
/// </summary>
/// <returns>tarifas rašant savam tinkle</returns>
public double namieSMSTarifas() { return NamieSMSTarifas; }
/// <summary>
/// Gražina kortelės žinučių tarifą rašant į kitą tinklą
/// </summary>
/// <returns>tarifas rašant į kitą tinklą</returns>
public double sveturSMSTarifas() { return SveturSMSTarifas; }
/// <summary>
/// Duomenų surikevimo užklojimas
/// </summary>
/// <returns>Surikiuoti duomenys</returns> public override
string ToString()
{
        string eilute;
        eilute = string.Format("| {0,11} | {1,4:f2} | {2,16:f2} | {3,17:f2} |
                                {4,13:f2} | {5,14:f2} |",
                                    Pavadinimas, PradineSum, NamieSkambuciuTarifas,
                                    SveturSkambuciuTarifas, NamieSMSTarifas, SveturSMSTarifas);
        return eilute;
}
/// <summary>
/// Patikrina, ar kortelė turi nemokamus skambučiu ir žinučių tarifus
        savam tinkle
/// </summary>
/// <param name="obj">reikšmė true arba false</param>
/// <returns>true arba false</returns> public override
bool Equals(object obj)
{
        return obj is Kortele kortele && this.NamieSkambuciuTarifas ==
                    0 && this.NamieSMSTarifas == 0 &&
                    kortele.NamieSkambuciuTarifas == 0 &&
                    kortele.NamieSMSTarifas == 0;
}
/// <summary>
/// Gražina Pavadinimo Hash kodą
/// </summary>
/// <returns>Pavadinimo Hash kodas</returns> public override
int GetHashCode()
{
         return -1054451132 + EqualityComparer<string>.Default.GetHashCode(Pavadinimas);
}
```

```
        /// <summary>
        /// Užklotas operatorius “<”
        /// </summary>
        /// <param name="kortele1">kortele 1</param>
        /// <param name="kortele2">kortele 2</param>
        /// <returns>true arba false</returns>
        public static bool operator <(Kortele kortele1, Kortele kortele2)
        {
            return kortele1.pradineSum() < kortele2.pradineSum() || (kortele1.pradineSum() ==
                    kortele2.pradineSum() &&
                   kortele1.pavadinimas().CompareTo(kortele2.pavadinimas()) < 0);
        }
        /// <summary>
        /// Užklotas operatorius >
        /// </summary>
        /// <param name="kortele1">kortele 1</param>
        /// <param name="kortele2">kortele 2</param>
        /// <returns>true arba false</returns>
        public static bool operator >(Kortele kortele1, Kortele kortele2)
        {
            return kortele1.pradineSum() > kortele2.pradineSum() || (kortele1.pradineSum() ==
                    kortele2.pradineSum() &&
                    kortele1.pavadinimas().CompareTo(kortele2.pavadinimas()) > 0);
        }


namespace P4_labaratorinis
{
    /// <summary>
    /// Konteinerinė klasė Tinklas
    /// </summary> internal class
    Tinklas
    {
        const int CN = 100;
        private Kortele[] korteles; private int
        Kiek;
        /// <summary>
        /// Konteinerinės klasės konstruktorius
        /// </summary> public
        Tinklas()
        {
            korteles = new Kortele[CN]; Kiek = 0;
        }
        /// <summary>
        /// Gražina konkretu klases Kortele elementa
        /// </summary>
        /// <param name="i">indeksas</param>
        /// <returns>Korteles itasis elementas</returns>
        public Kortele ImtiKonkretuElementa(int i) { return korteles[i]; }
        /// <summary>
        /// Gražina kiekį
        /// </summary>
        /// <returns>kiekis</returns> public int kiek() {
        return Kiek; }
        /// <summary>
        /// Ideda klasės elementą į konteinerį
        /// </summary>
        /// <param name="kortele">klasės Kortele elementas</param> public void
        Dėti(Kortele kortele) { korteles[Kiek++] = kortele; }
        /// <summary>
        /// Grąžina kortelę pagal nurodytą indeksą.
        /// </summary>
        /// <param name="kiek">Kortelės indeksas</param>
        /// <returns>Kortelė pagal nurodytą indeksą.</returns> public Kortele
```

```
        Indeksas(int kiek)
        {
            return korteles[kiek];
        }
        /// <summary>
        /// Sukeičia dviejų kortelių vietas masyve pagal nurodytus indeksus
        /// </summary>
        /// <param name="i">Pirmo elemento indeksas</param>
        /// <param name="j">Antro elemento indeksas</param> public void
        Keitimas(int i, int j)
        {
            Kortele laikinas = korteles[i]; korteles[i] =
            korteles[j]; korteles[j] = laikinas;
        }
        /// <summary>
        /// Surikiuoja elementus
        /// </summary>
        public void Rikevimas()
        {
            for (int i = 0; i < kiek() - 1; i++)
            {
                for (int j = i + 1; j < kiek(); j++)
                {
                    Kortele kortele1 = Indeksas(i); Kortele
                    kortele2 = Indeksas(j); if (kortele2 <
                    kortele1)
                    {
                        Keitimas(i, j);
                    }
                }
            }
        }
    }
}


using System.IO;
namespace P4_labaratorinis
{
    internal class Program
    {
        const int CN = 100;
        const string DF = "duom.txt";
        const string REZ = "..\\..\\rezultatai.txt"; const string LENTELE =
        "|                                        ----------------------------------------------------------------------
                                                  ------------------------------------------------------------
                                                                                                              |\n" +
                                              "| Pavadinimas | Suma | Skambuciai namie |
                                                 Skambučiai svetur | Žinutės namie | Žinutės
                                                 svetur |\n" +
                                              "| ------------------| ---------| -------------------------| -------------
                                                 --------------| ---------------------| ----------------------|";
        static void Main(string[] args)
        {
            Kortele[] korteles = new Kortele[CN]; Tinklas
            tinklas = new Tinklas();
            Tinklas maziausiasSveturSMS = new Tinklas(); Tinklas
            nemokami = new Tinklas();
            if(File.Exists(REZ))
                File.Delete(REZ);
            Nuskaitymas(ref tinklas, DF); Isvedimas(tinklas,
            "Duomenys");
            MaziausiasSveturSMS(tinklas, maziausiasSveturSMS); Isvedimas(maziausiasSveturSMS,
            "Maziausi SMS svetur"); NemokamuAtrinkimas(tinklas, nemokami);
            Isvedimas(nemokami, "Nemokami skambuciai ir SMS namuose"); nemokami.Rikevimas();
```

```
        Isvedimas(nemokami, "Surikiuoti nemokami skambuciai ir SMS namuose");
}
/// <summary>
/// Duomenų nuskaitymas
/// </summary>
/// <param name="tinklas">Konteineris tinklas</param>
/// <param name="Df">Duomenų failas</param>
static void Nuskaitymas(ref Tinklas tinklas, string Df)
{
        string pavadinimas;
        double pradineSum, namieSkambuciuTarifas, sveturSkambuciuTarifas, namieSMSTarifas,
                sveturSMSTarifas;
        using (StreamReader reader = new StreamReader(Df))
        {
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                        string[] parts = line.Split(' '); pavadinimas = parts[0];
                        pradineSum = double.Parse(parts[1]); namieSkambuciuTarifas =
                        double.Parse(parts[2]); sveturSkambuciuTarifas =
                        double.Parse(parts[3]); namieSMSTarifas = double.Parse(parts[4]);
                        sveturSMSTarifas = double.Parse(parts[5]);
                        Kortele kortele = new Kortele(pavadinimas, pradineSum, namieSkambuciuTarifas,
                                sveturSkambuciuTarifas, namieSMSTarifas, sveturSMSTarifas);
                                tinklas.Dėti(kortele);
                }
        }
}
/// <summary>
/// Tikrina ar duomenys teisingi
/// </summary>
/// <param name="tinklas">Konteineris tinklas</param>
/// <returns>true arba false</returns> static bool
Tikrinimas(Tinklas tinklas)
{
        for (int i = 0; i < tinklas.kiek(); i++)
        {
                if (tinklas.ImtiKonkretuElementa(i).namieSMSTarifas() >= 0 &&
                        tinklas.ImtiKonkretuElementa(i).sveturSMSTarifas() >= 0 &&
                        tinklas.ImtiKonkretuElementa(i).namieSkambuciuTarifas() >= 0 &&
                        tinklas.ImtiKonkretuElementa(i).sveturSkambuciuTarifas() >=
                        0 && tinklas.ImtiKonkretuElementa(i).pradineSum() >= 0)
                {
                        return true;
                }
        }
        return false;
}
/// <summary>
/// Isveda duomenis
/// </summary>
/// <param name="tinklas">Konteineris tinklas</param>
/// <param name="etikete">Lentelės etiketė</param> static void
Isvedimas(Tinklas tinklas, string etikete)
{
        using (var rez = System.IO.File.AppendText(REZ))
        {
                rez.WriteLine("{0,50}",etikete); rez.WriteLine(LENTELE);
                for (int i = 0; i < tinklas.kiek(); i++)
                {
                        if (Tikrinimas(tinklas) == true)
                        {
```

```
                        rez.WriteLine("{0}", tinklas.ImtiKonkretuElementa(i).ToString());
                    }
                    else
                    {
                        i = tinklas.kiek();
                        rez.WriteLine("| {0,50} {1,41}", "Klaida", "|");
                    }
                }
                rez.WriteLine("|    ------------------------------------------------------------------------
                     -----------------------------------------------------------|\n");
            }
        }
        /// <summary>
        /// Randa maziausio žinučių tarifo rašant į kitus tinklus indeksą
        /// </summary>
        /// <param name="tinklas">Konteineris tinklas</param>
        /// <returns></returns>
        static int MaziausioSveturSMSTarifoIndeksas(Tinklas tinklas)
        {
            int mazIndex = 0;
            double maz = tinklas.ImtiKonkretuElementa(0).sveturSMSTarifas(); for (int i = 0; i <
            tinklas.kiek(); i++)
            {
                if (tinklas.ImtiKonkretuElementa(i).sveturSMSTarifas() < maz)
                {
                    mazIndex = i;
                }
            }
            return mazIndex;
        }
        /// <summary>
        /// Sudeda maziausio žinučių tarifo rašant į kitus tinklus į naują konteinerį
        /// </summary>
        /// <param name="tinklas">Konteineris tinklas</param>
        /// <param name="maziausiasSveturSMS">Maziausio žinučių tarifo rašant į
                                  kitus tinklus konteineris</param> static void
        MaziausiasSveturSMS(Tinklas tinklas, Tinklas
                     maziausiasSveturSMS)
        {
            for (int i = 0; i < tinklas.kiek(); i++)
            {
                if (tinklas.ImtiKonkretuElementa(i).sveturSMSTarifas() ==
                    tinklas.ImtiKonkretuElementa(MaziausioSveturSMSTarifoIndeksas(
                    tinklas)).sveturSMSTarifas())
                {
                    maziausiasSveturSMS.Dėti(tinklas.ImtiKonkretuElementa(i));
                }
            }
        }
        /// <summary>
        /// Atrenka korteles, su kuriom galima rašyti ir skambinti savam tinkle
            nemokamai
        /// </summary>
        /// <param name="tinklas">Konteineris tinklas</param>
        /// <param name="nemokami">Konteineris nemokami</param>
        static void NemokamuAtrinkimas(Tinklas tinklas, Tinklas nemokami)
        {
            for (int i = 0; i < tinklas.kiek(); i++)
            {
                Kortele kortele = tinklas.ImtiKonkretuElementa(i); if
                (kortele.Equals(new Kortele()))
                {
                    nemokami.Dėti(kortele);
                }
```

```
                }
            }
        }
    }
```

3.3. Pradiniai duomenys ir rezultatai

Įprasti duomenįs:

```
Labas 30 0.5 0.7 0.0 0.5
Bite 35 0.0 0.7 0.2 0.5
Pildyk 20 0.5 0.7 0.2 0.5
Telia 32 0.0 0.5 0.0 0.7
Talia 20 0.0 0.3 0.0 0.0
```

```
                                        Duomenys
|-----------------------------------------------------------------------------------------------------------------------|
| Pavadinimas |  Suma | Skambuciai namie | Skambučiai svetur | Žinutės namie |   Žinutės svetur |
|-------------|-------|------------------|-------------------|---------------|------------------|
|       Labas | 30.00 |             0.50 |              0.70 |          0.00 |             0.50 |
|        Bite | 35.00 |             0.00 |              0.70 |          0.20 |             0.50 |
|      Pildyk | 20.00 |             0.50 |              0.70 |          0.20 |             0.50 |
|       Telia | 32.00 |             0.00 |              0.50 |          0.00 |             0.70 |
|       Talia | 20.00 |             0.00 |              0.30 |          0.00 |             0.00 |
|-----------------------------------------------------------------------------------------------------------------------|

                                    Maziausi SMS svetur
|-----------------------------------------------------------------------------------------------------------------------|
| Pavadinimas |  Suma | Skambuciai namie | Skambučiai svetur | Žinutės namie |   Žinutės svetur |
|-------------|-------|------------------|-------------------|---------------|------------------|
|       Talia | 20.00 |             0.00 |              0.30 |          0.00 |             0.00 |
|-----------------------------------------------------------------------------------------------------------------------|

                    Nemokami skambuciai ir SMS namuose
|-----------------------------------------------------------------------------------------------------------------------|
| Pavadinimas |  Suma | Skambuciai namie | Skambučiai svetur | Žinutės namie |   Žinutės svetur |
|-------------|-------|------------------|-------------------|---------------|------------------|
|       Telia | 32.00 |             0.00 |              0.50 |          0.00 |             0.70 |
|       Talia | 20.00 |             0.00 |              0.30 |          0.00 |             0.00 |
|-----------------------------------------------------------------------------------------------------------------------|

     Surikiuoti nemokami skambuciai ir SMS namuose
|-----------------------------------------------------------------------------------------------------------------------|
| Pavadinimas |  Suma | Skambuciai namie | Skambučiai svetur | Žinutės namie |   Žinutės svetur |
```

Talia	20.00	0.00	0.30	0.00	0.00
Telia	32.00	0.00	0.50	0.00	0.70

Duomenys neigiami:

```
Labas -30 -0.5 -0.7 -0.0 -0.5
Bite -35 -0.0 -0.7 -0.2 -0.5
Pildyk -20 -0.5 -0.7 -0.2 -0.5
Telia -32 -0.0 -0.5 -0.0 -0.7
Talia -20 -0.0 -0.3 -0.0 -0.0
```

Duomenys

Pavadinimas	Suma	Skambuciai namie	Skambučiai svetur	Žinutės namie	Žinutės svetur
Klaida					

Maziausi SMS svetur

Pavadinimas	Suma	Skambuciai namie	Skambučiai svetur	Žinutės namie	Žinutės svetur
Klaida					

Nemokami skambuciai ir SMS namuose

Pavadinimas	Suma	Skambuciai namie	Skambučiai svetur	Žinutės namie	Žinutės svetur
Klaida					

Surikiuoti nemokami skambuciai ir SMS namuose

Pavadinimas	Suma	Skambuciai namie	Skambučiai svetur	Žinutės namie	Žinutės svetur
Klaida					

3.4. Dėstytojo pastabos

5. Teksto analizė ir redagavimas

5.1. Darbo užduotis

U5-2. Nelyginis žodžių skaičius Tekstiniame faile pateikiamas tekstas. Žodžiai iš eilutės į kitą eilutę nekeliami. Žodžiai eilutėse skiriami bent vienu tarpu. Tarpai gali būti eilutės pradžioje bei gale, gali būti tuščios eilutės. Eilutėse, kuriose yra nelyginis žodžių skaičius n, n / 2 + 1 žodį pakeisti žodžiu „xxooxx“.

5.2. Programos tekstas

```
using System; using
System.IO; using
System.Linq;
using System.Text.RegularExpressions; namespace
P5_laboratorinis
{
    internal class Program
    {
        const string DF = "data.txt";
        const string REZ = "..\\..\\rez.txt"; const string AF =
        "..\\..\\analisis.txt"; static void Main(string[] args)
        {
                                    char[] separators = { ' ', ',', '.', '(', ')', '+', '-', '\\', '/',
                                                        '*', ':', '“', '„' };
            string[] lines; int[]
            wordCount; int
            lineCount;
            Scanning(DF, out lines, out wordCount, out lineCount, separators); using (StreamWriter af =
            new StreamWriter(AF))
            {
                af.WriteLine("Pakeisti žodžiai:");
                OddWordCount(ref lines, wordCount, lineCount, separators, af);
            }
            Results(REZ, lines, lineCount);
        }
        /// <summary>
        /// counts the words in one line
        /// </summary>
        /// <param name="line">text line</param>
        /// <param name="separators">separators</param>
        /// <returns>how many words are in one line</returns> static int
        WordCounting(string line, char[] separators)
        {
            string[] words = line.Split(separators,
                                    StringSplitOptions.RemoveEmptyEntries); return
            words.Length;
        }
        /// <summary>
        /// Scanning data from data file
        /// </summary>
        /// <param name="DF">data file</param>
        /// <param name="lines">text lines</param>
        /// <param name="wordCount">word count in one line</param>
        /// <param name="lineCount">number of lines</param>
        /// <param name="separators">separators</param>
        static void Scanning(string DF, out string[] lines, out int[] wordCount, out int lineCount, char[]
                                    separators)
        {
            lines = new string[1000]; wordCount = new
```

```
        int[1000];
        lineCount = 0;
        using (StreamReader df = new StreamReader(DF))
        {
            string line;
            while ((line = df.ReadLine()) != null)
            {
                line = line.Trim();
                if (!string.IsNullOrWhiteSpace(line))
                {
                    lines[lineCount] = line;
                    wordCount[lineCount] = WordCounting(line, separators); lineCount++;
                }
            }
        }
}
/// <summary>
/// changes lines which has odd number of words
/// </summary>
/// <param name="lines">text lines</param>
/// <param name="wordCount">number of words in one line</param>
/// <param name="lineCount">how many text lines are there</param>
/// <param name="separators">separators</param>
/// <param name="analysis">analysis file</param>
static void OddWordCount(ref string[] lines, int[] wordCount, int
                                    lineCount, char[] separators, StreamWriter analysis)
{
    for (int i = 0; i < lineCount; i++)
    {
        if (wordCount[i] % 2 > 0)
        {
            lines[i] = WordReplacement(lines[i], wordCount[i], separators,
                                                    analysis);
        }
    }
}
/// <summary>
/// replases the word
/// </summary>
/// <param name="line">text line</param>
/// <param name="wordCount">number of words in one line</param>
/// <param name="separators">separators</param>
/// <param name="analysis">analysis file</param>
/// <returns>line with a switchedword</returns>
static string WordReplacement(string line, int wordCount, char[]
                                        separators, StreamWriter analysis)
{
    string[] words = line.Split(separators,
                            StringSplitOptions.RemoveEmptyEntries); int
    replacedWordIndex = wordCount / 2;
    string replacedWord = words[replacedWordIndex]; words[replacedWordIndex] = "xxooxx";
    analysis.WriteLine(replacedWord);
    return SwitchedLine(line, words);
}
/// <summary>
/// forms a line with a switched word
/// </summary>
/// <param name="line">text line</param>
/// <param name="words">separated words in that line</param>
/// <returns>line with a switched word</returns>
static string SwitchedLine(string line, string[] words)
{
                string[] lineParts = Regex.Split(line, @"(\s+|[,\.\(\)\-
                                                    \+\\\/\*\:\!])");
```

```
            int wordIndex = 0;
            for (int i = 0; i < lineParts.Length; i++)
            {
                if (!string.IsNullOrWhiteSpace(lineParts[i]) && lineParts[i].All(char.IsLetterOrDigit))
                {
                    lineParts[i] = words[wordIndex]; wordIndex++;
                }
            }
            return string.Join("", lineParts);
        }
        /// <summary>
        /// writes the whole text but with switched words
        /// </summary>
        /// <param name="REZ">rezults file</param>
        /// <param name="lines">text lines</param>
        /// <param name="lineCount">number of lines</param>
        static void Results(string REZ, string[] lines, int lineCount)
        {
            using (StreamWriter rez = new StreamWriter(REZ))
            {
                for (int i = 0; i < lineCount; i++)
                {
                    rez.WriteLine(lines[i]);
                }
            }
        }
    }
}
```

5.3. Pradiniai duomenys ir rezultatai

Duomenys be tuščių eilučių:

```
a b                ,,,,,,,,,.,,,,,,, c              ,,, , , , ,,
adafa awda
awdd awdd awds
```

Rezultatų failas:

```
a xxooxx               ,,,,,,,,,.,,,,,,, c              ,,, , , , ,,
adafa awda
awdd xxooxx awds
```

Analizės failas:

```
Pakeisti žodžiai:
b
awdd
```

Duomenys su tuščiom eilutėm;

```
a b                ,,,,,,,,,.,,,,,,, c              ,,, , , , ,,

adafa awda

awdd awdd awds
```

Rezultatų failas:

```
a xxooxx                ,,,,,,,,,.,,,,,,,, c              ,,, , , , ,,
adafa awda
awdd xxooxx awds
```

Analizės failas:

```
Pakeisti žodžiai:
b
awdd
```

5.4. Dėstytojo pastabos

6. Susieti rinkiniai

6.1. Darbo užduotis

U6–2. Futbolas Pirmoje failo eilutėje nurodytas futbolo komandų skaičius. Tolesnėse eilutėse pateikta informacija apie futbolo komandas: pavadinimas, miestas, trenerio pavardė, vardas. Žemiau pateikta I rato rezultatų lentelė, išreikšta pelnytais įvarčiais. Suskaičiuokite kiekvienos komandos surinktų taškų skaičių, jei už pergalę skiriami 3 taškai, o už lygiąsias – 1 taškas. Sudarykite komandų turnyrinę lentelę – surikiuokite surinktų taškų mažėjimo tvarka. Jei komandos surinko taškų vienodai, aukščiau ta komanda, kuri turi daugiau pergalių. Suraskite daugiausiai įvarčių pelniusią komandą. Suraskite komandas, kurios daugiausiai rungtynių nepraleido įvarčių.

6.2. Programos tekstas

```
using System.IO;
namespace Dvimatis_masyvas_Laboras
{
    /// <summary>
    /// Team class
    /// </summary>
    class Team
    {
        /// <summary>
        /// Team's name
        /// </summary>
        public string Name { get; private set; }
        /// <summary>
        /// Team's city
        /// </summary>
        public string City { get; private set; }
        /// <summary>
        /// Team's coach's surname
        /// </summary>
        public string CoachSurname { get; private set; }
        /// <summary>
        /// Team's coach's name
        /// </summary>
        public string CoachName { get; private set; }
        /// <summary>
        /// constructor
        /// </summary>
        /// <param name="name">name</param>
        /// <param name="city">city</param>
        /// <param name="coachSurname">coach's surname</param>
        /// <param name="coachName">coach's name</param>
        public Team(string name, string city, string coachSurname, string coachName)
        {
            Name = name;
            City = city;
            CoachSurname = coachSurname;
            CoachName = coachName;
        }
        /// <summary>
        /// For sorted output
        /// </summary>
        /// <returns>Sorted line</returns>
        public override string ToString()
        {
            return $"{Name,-10} {City,-10} {CoachSurname,-15} {CoachName,-10}";
```

```
    }
}
/// <summary>
/// Team's conteiner
/// </summary>
class TeamsContainer
{
    private const int MaxTeams = 100;
    private Team[] teams;
    private int teamCount;
    /// <summary>
    /// Construtor
    /// </summary>
    public TeamsContainer()
    {
        teams = new Team[MaxTeams];
        teamCount = 0;
    }
    /// <summary>
    /// Returns team's count
    /// </summary>
    /// <returns>team's count</returns>
    public int TeamCount() { return teamCount; }
    /// <summary>
    /// Adds teams in it's conteiner
    /// </summary>
    /// <param name="team">team</param>
    public void AddTeam(Team team)
    {
        teams[teamCount++] = team;
    }
    /// <summary>
    /// Returns a team from a specific index
    /// </summary>
    /// <param name="index">index</param>
    /// <returns>team from a specific index</returns>
    public Team GetTeam(int index)
    {
        return teams[index];
    }
    /// <summary>
    /// Inserst team in a specific index
    /// </summary>
    /// <param name="index">index</param>
    /// <param name="team">team</param>
    public void TeamInsert(int index, Team team)
    {
        teams[index] = team;
    }
    /// <summary>
    /// For sorted output
    /// </summary>
    /// <returns>Sorted lines</returns>
    public override string ToString()
    {
        string result = "Name       City       Coach Surname   Coach Name\n";
        result += new string('-', 50) + "\n";
        for (int i = 0; i < teamCount; i++)
        {
            result += teams[i].ToString() + "\n";
```

```
        }
        return result;
    }
}
/// <summary>
/// Result's matrix
/// </summary>
class ResultsMatrix
{
    const int LINE = 1000;
    const int COLUMN = 1000;
    private int[,] A;
    /// <summary>
    /// Team goals
    /// </summary>
    public int TeamGoals { get; private set; }
    /// <summary>
    /// Team lines number
    /// </summary>
    public int TeamNum { get; private set; }
    /// <summary>
    /// Construktor
    /// </summary>
    public ResultsMatrix()
    {
        TeamGoals = 0;
        TeamNum = 0;
        A = new int[LINE, COLUMN];
    }
    /// <summary>
    /// Assigns a value to a specific matrix index
    /// </summary>
    /// <param name="lineIndex">line index</param>
    /// <param name="columnIndex">column index</param>
    /// <param name="goals"> value</param>
    public void Input(int lineIndex, int columnIndex, int goals)
    {
        A[lineIndex, columnIndex] = goals;
    }
    /// <summary>
    /// Returns a value in the specific index
    /// </summary>
    /// <param name="lineIndex">line index</param>
    /// <param name="columnIndex">column index</param>
    /// <returns>value</returns>
    public int GetValue(int lineIndex, int columnIndex)
    {
        return A[lineIndex, columnIndex];
    }
}
internal class Program
{
    const string DF = "..\\..\\data.txt";
    const string REZ = "..\\..\\rez.txt";
    static void Main(string[] args)
    {
        TeamsContainer teams = new TeamsContainer();
        ResultsMatrix results = new ResultsMatrix();
        Scanning(DF, ref teams, ref results);
        if (File.Exists(REZ))
```

```
            File.Delete(REZ);
        PointsCounting(ref results, teams, out int[] points, out int[] teamWins);
        Output(REZ, teams, results, points, "Original data:");
        if (DataChecking(results, teams))
        {
            Sorting(points, teamWins, teams, results);
            Output(REZ, teams, results, points, "Sorted data:");
            BestAttack(results, out string[] bestAttackers, teams);
            BestAtackDefenceTeamsOutput(REZ, "Teams who scored the most goals:", bestAttackers);
            BestDefence(results, out string[] bestDefenders, teams);
            BestAtackDefenceTeamsOutput(REZ, "Teams who had the most games without conceding goals:",
                                                    bestDefenders);
        }
        else
        {
            using (var rez = File.AppendText(REZ))
            {
                rez.WriteLine("Error");
            }
        }
    }
    /// <summary>
    /// Data sacanning
    /// </summary>
    /// <param name="DF">data file</param>
    /// <param name="teams"> teams conteiner</param>
    /// <param name="results"> results matrix</param>
    static void Scanning(string DF, ref TeamsContainer teams, ref ResultsMatrix results)
    {
        using (StreamReader df = new StreamReader(DF))
        {
            int teamCount = int.Parse(df.ReadLine());
            for (int i = 0; i < teamCount; i++)
            {
                string[] parts = df.ReadLine().Split(' ');
                string name = parts[0];
                string city = parts[1];
                string coachSurname = parts[2];
                string coachName = parts[3];
                Team team = new Team(name, city, coachSurname, coachName);
                teams.AddTeam(team);
            }
            for (int i = 0; i < teamCount; i++)//line
            {
                string[] line = df.ReadLine().Split(' ');
                for (int j = 0; j < teamCount; j++)//column
                {
                    results.Input(i, j, int.Parse(line[j]));
                }
            }
        }
    }
    /// <summary>
    /// Checks is the data correct
    /// </summary>
    /// <param name="results">results matrix</param>
    /// <param name="teams">teams conteiner</param>
    /// <returns>false if data is incorrect, true if data is correct</returns>
    static bool DataChecking(ResultsMatrix results, TeamsContainer teams)
    {
```

```
    bool correctData = false;
    for (int i = 0; i < teams.TeamCount(); i++)
    {
        for (int j = 0; j < teams.TeamCount(); j++)
        {
            if (results.GetValue(i, j) < 0)
            {
                correctData = false;
                return correctData;
            }
            else correctData = true;
        }
    }
    return correctData;
}
/// <summary>
/// Data output
/// </summary>
/// <param name="REZ">result's file</param>
/// <param name="teams">team's conteiner</param>
/// <param name="results">result's matrix</param>
/// <param name="points">team's points</param>
/// <param name="title">title</param>
static void Output(string REZ, TeamsContainer teams, ResultsMatrix results, int[] points, string title)
{
    using (var rez = File.AppendText(REZ))
    {
        rez.WriteLine(title);
        rez.WriteLine();
        rez.WriteLine(teams.ToString());
        rez.WriteLine("Results:");
        if (DataChecking(results, teams))
        {
            for (int i = 0; i < teams.TeamCount(); i++)
            {
                for (int j = 0; j < teams.TeamCount(); j++)
                {
                    rez.Write("{0}   ", results.GetValue(i, j));
                }
                rez.WriteLine("  Score: {0}", points[i]);
            }
        }
        else
        {
            for (int i = 0; i < teams.TeamCount(); i++)
            {
                for (int j = 0; j < teams.TeamCount(); j++)
                {
                    rez.Write("{0}   ", results.GetValue(i, j));
                }
                rez.WriteLine();
            }
        }
        rez.WriteLine();
        rez.WriteLine();
    }
}
/// <summary>
/// List of team's who scored the most goals and teams who had the most games without consending goals
/// </summary>
/// <param name="REZ">result's file</param>
/// <param name="title">title</param>
```

```
/// <param name="bestTeam">best team at something</param>
static void BestAtackDefenceTeamsOutput(string REZ, string title, string[] bestTeam)
{
    using (var rez = File.AppendText(REZ))
    {
        rez.WriteLine(title);
        rez.WriteLine();
        for (int i = 0; i < bestTeam.Length; i++)
        {
            if (!string.IsNullOrWhiteSpace(bestTeam[i]))
            {
                rez.WriteLine(bestTeam[i]);
            }
        }
        rez.WriteLine();
        rez.WriteLine();
    }
}
/// <summary>
/// Counts team's points
/// </summary>
/// <param name="results">result's matrix</param>
/// <param name="teams">team's conteiner</param>
/// <param name="points">team's points</param>
/// <param name="teamWins">team's wins</param>
static void PointsCounting(ref ResultsMatrix results, TeamsContainer teams, out int[] points, out int[]
                                    teamWins)
{
    points = new int[teams.TeamCount()];
    teamWins = new int[teams.TeamCount()];
    for (int i = 0; i < teams.TeamCount(); i++)
    {
        for (int j = 0; j < teams.TeamCount(); j++)
        {
            if (DataChecking(results, teams))
            {
                if (i != j)
                {
                    if (results.GetValue(j, i) < results.GetValue(i, j))
                    {
                        points[i] = points[i] + 3;
                        teamWins[i]++;
                    }
                    else if (results.GetValue(j, i) ==results.GetValue(i,j))
                    {
                        points[i] = points[i] + 1;
                    }
                    else
                    {
                        ;
                    }
                }
            }
            else
            {
                return;
            }
        }
    }
}
```

```
/// <summary>
/// Sorts data
/// </summary>
/// <param name="points">team's points</param>
/// <param name="teamWins">team's wins</param>
/// <param name="teams">team's conteiner</param>
/// <param name="results">result's matrix</param>
static void Sorting(int[] points, int[] teamWins, TeamsContainer teams, ResultsMatrix results)
{
   int teamCount = teams.TeamCount();
   for (int i = 0; i < teamCount - 1; i++)
   {
      for (int j = i + 1; j < teamCount; j++)
      {
         if (points[i] < points[j]
            || (points[i] == points[j] && teamWins[i] < teamWins[j])
            || (points[i] == points[j] && teamWins[i] == teamWins[j] && i > j))
         {
            int tempPoints = points[i];
            points[i] = points[j];
            points[j] = tempPoints;
            int tempWins = teamWins[i];
            teamWins[i] = teamWins[j];
            teamWins[j] = tempWins;
            Team tempTeam = teams.GetTeam(i);
            teams.TeamInsert(i, teams.GetTeam(j));
            teams.TeamInsert(j, tempTeam);
            for (int k = 0; k < teamCount; k++)
            {
               int tempRowValue = results.GetValue(i, k);
               results.Input(i, k, results.GetValue(j, k));
               results.Input(j, k, tempRowValue);
            }
            for (int k = 0; k < teamCount; k++)
            {
               int tempColumnValue = results.GetValue(k, i);
               results.Input(k, i, results.GetValue(k, j));
               results.Input(k, j, tempColumnValue);
            }
         }
      }
   }
}
/// <summary>
/// Finds teams which scored the most goals
/// </summary>
/// <param name="results">results matrix</param>
/// <param name="bestAttackers">the best attacking team's names</param>
/// <param name="teams">teams conteiner</param>
static void BestAttack(ResultsMatrix results, out string[] bestAttackers, TeamsContainer teams)
{
   int[] teamGoals = new int[teams.TeamCount()];
   int mostGoals = 0, nr = 0;
   bestAttackers = new string[teams.TeamCount()];
   for (int i = 0; i < teams.TeamCount(); i++)
   {
      for (int j = 0; j < teams.TeamCount(); j++)
      {
         teamGoals[i] += results.GetValue(i, j);
      }
```

```
            if (teamGoals[i] > mostGoals)
            {
                mostGoals = teamGoals[i];
                bestAttackers = new string[teams.TeamCount()];
                nr = 0;
            }
            if (teamGoals[i] == mostGoals)
            {
                bestAttackers[nr] = teams.GetTeam(i).Name;
                nr++;
            }
        }
    }
    /// <summary>
    /// Finds teams which had the most games without consending goals
    /// </summary>
    /// <param name="results">results matrix</param>
    /// <param name="bestDefenders">the best defending team's names</param>
    /// <param name="teams">teams conteiner</param>
    static void BestDefence(ResultsMatrix results, out string[] bestDefenders, TeamsContainer teams)
    {
        bestDefenders = new string[teams.TeamCount()];
        int[] allGoalsDefended = new int[teams.TeamCount()];
        int mostSaves = 0;
        int nr = 0;
        for (int j = 0; j < teams.TeamCount(); j++)
        {
            for (int i = 0; i < teams.TeamCount(); i++)
            {
                if (i != j)
                {
                    if (results.GetValue(i, j) == 0)
                    {
                        allGoalsDefended[j]++;
                    }
                }
            }
            if (allGoalsDefended[j] > mostSaves)
            {
                mostSaves = allGoalsDefended[j];
                bestDefenders = new string[teams.TeamCount()];
                nr = 0;
            }
            if (allGoalsDefended[j] == mostSaves)
            {
                bestDefenders[nr] = teams.GetTeam(j).Name;
                nr++;
            }
        }
    }
}
```

6.3. Pradiniai duomenys ir rezultatai

Įprasti duomenys:

```
4
Komanda1 Vilnius Petrauskas Jonas
```

```
Komanda2 Kaunas Jankauskas Tomas
Komanda3 Klaipeda Kazlauskas Povilas
Komanda4 Siauliai Stasys Antanas
0 1 3 1
2 0 2 4
0 0 0 2
0 3 3 0
```

Rezultatai:

```
Original data:


Name       City       Coach Surname   Coach Name
-------------------------------------------------
Komanda1   Vilnius    Petrauskas      Jonas
Komanda2   Kaunas     Jankauskas      Tomas
Komanda3   Klaipeda   Kazlauskas      Povilas
Komanda4   Siauliai   Stasys          Antanas

Results:
0   1   3   1     Score: 6
2   0   2   4     Score: 9
0   0   0   2     Score: 0
0   3   3   0     Score: 3



Sorted data:

Name       City       Coach Surname   Coach Name
-------------------------------------------------
Komanda2   Kaunas     Jankauskas      Tomas
Komanda1   Vilnius    Petrauskas      Jonas
Komanda4   Siauliai   Stasys          Antanas
Komanda3   Klaipeda   Kazlauskas      Povilas

Results:
0   2   4   2     Score: 9
1   0   1   3     Score: 6
3   0   0   3     Score: 3
0   0   2   0     Score: 0



Teams who scored the most goals:
```

```
Komanda2



Teams who had the most games without conceding goals:

Komanda1
```

Duomenys neigiami:

```
4
Komanda1 Vilnius Petrauskas Jonas
Komanda2 Kaunas Jankauskas Tomas
Komanda3 Klaipeda Kazlauskas Povilas
Komanda4 Siauliai Stasys Antanas
-1 1 3 1
2 0 2 4
0 0 0 2
0 3 3 0
```

Rezultatai:

Original data:

Name City Coach Surname Coach Name

Komanda1 Vilnius Petrauskas Jonas

Komanda2 Kaunas Jankauskas Tomas

Komanda3 Klaipeda Kazlauskas Povilas

Komanda4 Siauliai Stasys Antanas

Results:

-1 1 3 1 Score: 0

2 0 2 4 Score: 0

0 0 0 2 Score: 0

0 3 3 0 Score: 0

Error

6.4. Dėstytojo pastabos