```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace _1_egzo_kl
{
        public class Asmuo
        {
            public string pavVard { get; set; } // pavardė, vardas
            public int amžius { get; set; } // amžius
            public TimeSpan laikas { get; set; } // atvykimo laikas
            public Asmuo(string pavVard, int amžius, TimeSpan laikas)
            {
                this.pavVard = pavVard;
                this.amžius = amžius;
                this.laikas = laikas;
            }
            public override string ToString()
            {
                string eilute;
                eilute = string.Format("{0, -17} {1} {2}", pavVard, amžius, laikas);
                return eilute;
            }

            // Užklotas metodas GetHashCode()
            public override int GetHashCode()
            {
                return base.GetHashCode();
            }
        }
        class Program
        {
            const string CFd1 = "..\\..\\Asmenys.txt"; // asmenų duomenų failo vardas
            static void Main(string[] args)
            {
            int metai = 0;
            int k = 0;
                // Asmenų sąrašo sudarymas ir spausdinimas
                List<Asmuo> AsmenuList = SkaitytiAsmuoList(CFd1);
                SpausdintiAsmenuList(AsmenuList, "Pradiniai duomenys");
                Queue<Asmuo> Eile = new Queue<Asmuo>();
            SortedDictionary<int, Asmuo> zodynas = new SortedDictionary<int, Asmuo>();
                TimeSpan atvykimoPradzia = new TimeSpan(7, 0, 0); // tikrinamo laiko
intervalo pradžia
                TimeSpan atvykimoPabaiga = new TimeSpan(10, 0, 0); // tikrinamo laiko
intervalo pabaiga
                TimeSpan žingsnis = new TimeSpan(0, 10, 0); // laiko intervalo peržiūros
žingsnis
                int[] Raktai = { 7, 9, 4, 1, 6, 2, 3 }; // žodyno raktai
                                                        // ATLIKITE: visus nurodytus
skaičiavimus
                Atrinkti(AsmenuList, Eile, atvykimoPradzia, atvykimoPabaiga, žingsnis);
                Console.WriteLine("Eiles elementu kiekis:{0}",Eile.Count());

            foreach (Asmuo a in Eile)
```

```
            {
                if (k == 0)
                {
                    Console.WriteLine("Pirmas eiles elementas:{0}", a);
                    k++;
                    metai = a.amžius;
                }
            }
            Formuoti(AsmenuList, zodynas, metai, Raktai);
            Spausdinti(zodynas);
            double sum = 0;
            double average;
            foreach(KeyValuePair<int,Asmuo> p in zodynas)
            {
               sum =  p.Value.amžius + sum;
            }
            average = sum / zodynas.Count();
            Console.WriteLine("Vidurkis: {0}", average);
            }
            // spausdina asmenų duomenų lentelę
            static void SpausdintiAsmenuList(List<Asmuo> AsmuoList, string antraste)
            {
                const string virsus =
                "------------------------------------------------ \r\n"
                + " Nr. Pavardė, vardas Amžius Atvykimo laikas \r\n"
                + "------------------------------------------------";
                Console.WriteLine("\n " + antraste);
                Console.WriteLine(virsus);
                for (int i = 0; i < AsmuoList.Count; i++)
                {
                    Asmuo zmog = AsmuoList[i];
                    Console.WriteLine("{0, 3} {1}", i + 1, zmog);
                }
                Console.WriteLine("------------------------------------------------
\n");
            }

            // skaito asmenų duomenų failą
            static List<Asmuo> SkaitytiAsmuoList(string fv)
            {
                // asmenų dinaminis masyvas
                List<Asmuo> AsmuoList = new List<Asmuo>();
                using (StreamReader srautas = new StreamReader(fv,
Encoding.GetEncoding(1257)))
                {
                    string eilute;
                    while ((eilute = srautas.ReadLine()) != null)
                    {
                        string[] eilDalis = eilute.Split(';');
                        string pav = eilDalis[0];
                        int amžius = int.Parse(eilDalis[1]);
                        TimeSpan laikas = TimeSpan.Parse(eilDalis[2]);
                        Asmuo naujas = new Asmuo(pav, amžius, laikas);
                        AsmuoList.Add(naujas);
                    }
                }
                return AsmuoList;
            }
```

```
            // spausdina žodyno reikšmes
            // naudoja ENumerator
            public static void Spausdinti(SortedDictionary<int, Asmuo> zodynas)
            {
                var enumerator = zodynas.GetEnumerator();
                while (enumerator.MoveNext())
                {
                    object item = enumerator.Current;
                    Console.WriteLine(" {0} ", item);
                }
                Console.WriteLine();
            }
            // Formuoja eilę
            static void Atrinkti(List<Asmuo> AsmenuList, Queue<Asmuo> Eile,
            TimeSpan atvykimoPradzia, TimeSpan atvykimoPabaiga,
            TimeSpan žingsnis)
            {
                // ATLIKITE: Dinaminio masyvo asmenys, kurių atvykimo laikas yra duotame
                // intervale [atvykimoPradzia, atvykimoPradzia] ir yra kartotinis duotam
                // žingsniui žingsnis, įrašomi į eilės konteinerį.
                for(int i = 0;i< AsmenuList.Count();i++)
                {

                      Asmuo duom = AsmenuList[i];
                      var dalyba = duom.laikas.TotalMinutes / žingsnis.TotalMinutes;
                      if ((duom.laikas.TotalMinutes >= atvykimoPradzia.TotalMinutes) &&
(duom.laikas.TotalMinutes  <= atvykimoPabaiga.TotalMinutes) && (dalyba != 0))
                        {
                            Eile.Enqueue(duom);
                        }
                }
            }
            // Formuoja žodyną
            static void Formuoti(List<Asmuo> AsmenuList, SortedDictionary<int, Asmuo>
Zodynas,
            int metai, int[] Raktai)
            {
                // ATLIKITE: Dinaminio masyvo asmenys, kurių amžius didesnis už duotą
sveiką skaičių
                // metai, surašomi į rikiuotą žodyną. Žodyno raktai – sveiki skaičiai,
paeiliui imami
                // iš duoto sveikų skaičių masyvo Raktai, o reikšmės – Asmuo klasės
objektai.
                for(int i = 0; i < AsmenuList.Count();i++)
                {
                Asmuo duom = AsmenuList[i];
                    if(duom.amžius > metai)
                    {
                        Zodynas.Add(Raktai[i], duom);
                    }
                }
            }
        }
    }
```

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

namespace _2dalis
{
    abstract class Kandidatas : Object
    {
        protected const double BazinisDydis = 800.00; // Bazinis atlyginimo dydis
        protected string PavVrd { get; set; } // Pavardė ir vardas
        protected int Amzius { get; set; } // Amžius
        protected double Stažas { get; set; } // Darbo stažas (metais)
                                              // Klasės konstruktorius
        public Kandidatas(string PavVrd = "", int Amzius = 0, double Stažas = 0.0)
        {
            this.PavVrd = PavVrd;
            this.Amzius = Amzius;
            this.Stažas = Stažas;
        }
        // Abstraktus metodas
        public abstract double Atlyginimas();
        public override string ToString()
        {
            // ATLIKITE: Užklokite kandidato spausdinimo eilute metodą
            string eilute;
            eilute = string.Format("{0}  {1, -10}   {2}
{3}",BazinisDydis,PavVrd,Amzius,Stažas);
            return eilute;
        }
    }
    class Programuotojas : Kandidatas
    {
        // ATLIKITE: Aprašykite klasės savybes ir konstruktorių
        // ATLIKITE: Užklokite programuotojo atlyginimo skaičiavimo metodą
        // ATLIKITE: Užklokite programuotojo spausdinimo eilute metodą

        public double patirtis { get; set; }
        public int nusiskundimai { get; set; }
        public Programuotojas(string pavvrd = "", int amzius = 0, double stazas = 0,
double patirtis = 0, int nusiskundimai = 0) : base(pavvrd,amzius,stazas)
        {
            this.patirtis = patirtis;
            this.nusiskundimai = nusiskundimai;
        }
        public override string ToString()
        {
            string eilute;
            eilute = string.Format("{0}  {1}  {2}", base.ToString(), patirtis,
nusiskundimai);
            return eilute;
        }
        public override double Atlyginimas()
        {
            return BazinisDydis + ((0.2 * BazinisDydis) * (1.1 * patirtis)) + ((0.1 *
BazinisDydis) * (nusiskundimai * -1));
        }
```

```
    }
    class Program
    {
        static void Main(string[] args)
        {
            // Programuotojų objektų masyvas P(n)
            int n = 3; Programuotojas[] P = new Programuotojas[n];
            // P(n) objektų užpildymas reikšmėmis
            P[0] = new Programuotojas("Programuotojas1", 29, 1.1, 1.5, 0);
            P[1] = new Programuotojas("Programuotojas2", 39, 11.5, 2.2, 3);
            P[2] = new Programuotojas("Programuotojas3", 30, 3.0, 3.6, 0);
            // ATLIKITE: Papildykite Main metodą reikiamais veiksmais
            Spausdinti(P, n);
        }
        public static void Spausdinti(Kandidatas[] K, int kn)
        {
            // ATLIKITE: Spausdinkite kiekvieno kandidato atlyginimą ekrane
            for(int i = 0; i < kn;i++)
            {
                Console.WriteLine("Nr. {0} {1}", i + 1, K[i]);
                Console.WriteLine("Atlyginimas {0}", K[i].Atlyginimas());
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace _3dalis
{
    public class Knyga
    {
        public string Pavadinimas { get; set; }
        public int Metai { get; set; }
        public int PuslapiuSk { get; set; }
        public Knyga(string pavad = "", int metai = 0, int puslSk = 0)
        {
            this.Pavadinimas = pavad;
            this.Metai = metai;
            this.PuslapiuSk = puslSk;
        }
        public override string ToString()
        {
            string eilute;
            eilute = string.Format("{0, -20} {1, 4:d} {2, 4:d}",
            Pavadinimas, Metai, PuslapiuSk);
            return eilute;
        }
    }
    // Klasė sąrašo vienam elementui saugoti
    public sealed class Mazgas
    {
        public Knyga Duom { get; set; } // duomenys
        public Mazgas Kitas { get; set; } // nuoroda į kitą elementą
        public Mazgas(Knyga duom, Mazgas adresas)
        {
            this.Duom = duom;
            this.Kitas = adresas;
        }
    }
    // Klasė knygų duomenims saugoti
    public sealed class Sąrašas
    {
        private Mazgas pr; // sąrašo pradžia
        private Mazgas pb; // sąrašo pabaiga
        private Mazgas ss; // sąrašo sąsaja
                           // Pradinės sąrašo nuorodų reikšmės
        public Sąrašas()
        {
            this.pr = null;
            this.pb = null;
            this.ss = null;
        }
        // Grąžina sąrašo sąsajos elemento reikšmę (duomenis)
        public Knyga ImtiDuomenis() { return ss.Duom; }
        // Sukuriamas sąrašo elementas ir prijungiamas prie sąrašo pabaigos
        // nauja – naujo elemento reikšmė (duomenys)
        public void DėtiDuomenisT(Knyga nauja)
        {
```

```
            // ATLIKITE: padėkite naują elementą sąrašo pabaigoje
            var ss = new Mazgas(nauja, null);
            if(pr!=null)
            {
                pb.Kitas = ss;
                pb = ss;
            }
            else
            {
                pr = ss;
                pb = ss;
            }
        }
        // Sąsajai priskiriama sąrašo pradžia
        public void Pradžia() { ss = pr; }
        // Sąsajai priskiriamas sąrašo sekantis elementas
        public void Kitas() { ss = ss.Kitas; }
        // Grąžina true, jeigu sąsaja netuščia
        public bool Yra() { return ss != null; }
        // Šalina sąsajos rodomą elementą
        public void Salinti()
        {
            if (ss == null) return;
            if(ss == pr)
            {
                pr = pr.Kitas;
                ss = pr;
            }
            else
            {
                Mazgas v = pr;
                while (v.Kitas != ss)
                    v = v.Kitas;
                v.Kitas = ss.Kitas;
                ss = v;
            }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Sąrašas knygos = new Sąrašas();
            const string CFd = @"..\..\Knygos.txt";
            const string CFr = @"..\..\Rezultatai.txt";
            if (File.Exists(CFr))
                File.Delete(CFr);
            // ATLIKITE: skaitykite duomenis iš failo į tiesioginį sąrašą Knygos,
            // spausdinkite duomenis, pašalinkite knygas,
            // kurių pavadinime yra daugiau nei nurodytas knygų skaičius,
            // spausdinkite sąrašą.
            int skaicius;
            Console.WriteLine("Iveskite knygos pavadinimo zodziu skaiciu");
            skaicius = int.Parse(Console.ReadLine());
            knygos = ĮvestiTiesiog(CFd);
            Spausdinti(CFr, knygos, "Pradiniai duomenys");
            Console.WriteLine("a");
            Išmesti(knygos, skaicius);
```

```
            Console.WriteLine("aa");
            Spausdinti(CFr, knygos, "Atrinktos");
            Console.WriteLine("aaa");
            Console.WriteLine("Programa darbą baigė.");

        }
        // Skaitomos objektų reikšmės iš failo ir sudedamos į sąrašą tiesiogine tvarka
        // fv – duomenų failo vardas
        // Grąžina - sąrašo objekto nuorodą
        static Sąrašas ĮvestiTiesiog(string fv)
        {
            Sąrašas A = new Sąrašas();
            using (var failas = new StreamReader(fv, Encoding.GetEncoding(1257)))
            {
                string pavad;
                int metai;
                int pslSk;
                string eilute;
                while ((eilute = failas.ReadLine()) != null)
                {
                    var eilDalis = eilute.Split(';');
                    pavad = eilDalis[0];
                    metai = Convert.ToInt32(eilDalis[1]);
                    pslSk = Convert.ToInt32(eilDalis[2]);
                    var Knyga = new Knyga(pavad, metai, pslSk);
                    A.DėtiDuomenisT(Knyga);
                }
            }
            return A;
        }
        // Sąrašo A duomenys spausdinami lentele faile fv
        // fv – duomenų failo vardas
        // A - sąrašo objekto adresas
        // antraste - lentelės pavadinimas
        static void Spausdinti(string fv, Sąrašas A, string antraste)
        {
            using (var failas = new StreamWriter(fv, true))
            {
                failas.WriteLine(antraste);
                failas.WriteLine("-------------------------------------");
                failas.WriteLine("Pavadinimas Metai Puslapiai ");
                failas.WriteLine("-------------------------------------");
                // Sąrašo peržiūra, panaudojant sąsajos metodus
                for (A.Pradžia(); A.Yra(); A.Kitas())
                {
                    failas.WriteLine(A.ImtiDuomenis());
                }
                failas.WriteLine("-------------------------------------\n");
            }
        }
        // Iš sąrašo išmeta knygas, kurių pavadinime yra didesnis nei nurodytas žodžių
skaičius
        // A - sąrašo objekto adresas
        // zodSkaicius - žodžių skaičius knygos pavadinime
        static void Išmesti(Sąrašas A, int zodSkaicius)
        {
            // ATLIKITE: pašalinkite iš sąrašo knygas, kurių pavadinime yra didesnis nei
            // nurodytas žodžių skaičius
```

```
            for(A.Pradžia();A.Yra(); A.Kitas())
            {
                int n = 0;
                foreach(var a in A.ImtiDuomenis().Pavadinimas.Split(' '))
                {
                    n++;
                }
                if (n >= zodSkaicius)
                {
                    A.Salinti();
                }
            }
        }
    }
}
```