

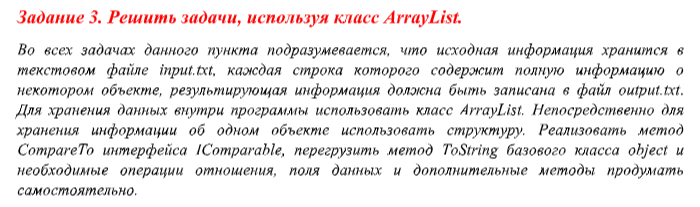


Stack<char> words = new Stack<char>();  
string vowels = "аяуюоеёэиы";  
char letter;  
using (StreamWriter writer = new StreamWriter("Task1.txt"))  
{  
 writer.WriteLine("Привет, как дела?");  
}  
  
using (StreamReader reader = new StreamReader("Task1.txt"))  
{  
 while (!reader.EndOfStream)  
 {  
 letter = (char)reader.Read();  
 if (vowels.Contains(letter))  
 words.Push(letter);  
 }  
}  
  
Console.WriteLine(string.Join(" ", words));



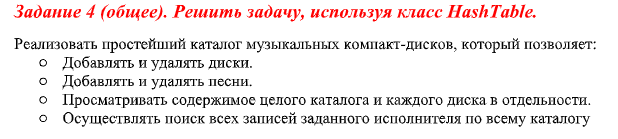


double count;  
double a = 1;  
double b = 5;  
Queue<double> AandB = new Queue<double>();  
Queue<double> MinA = new Queue<double>();  
Queue<double> MoreB = new Queue<double>();  
  
using (StreamReader reader = new StreamReader("Task2.txt"))  
{  
 while (!reader.EndOfStream)  
 {  
 count = Convert.*ToInt32*(reader.ReadLine());  
 if (count < a)  
 MinA.Enqueue(count);  
 else if (count > b)  
 MoreB.Enqueue(count);  
 else  
 AandB.Enqueue(count);  
 }  
}  
  
Console.*WriteLine*($"Числа в диапазоне от A До B: {String.*Join*(" ", AandB)}");  
Console.*WriteLine*($"Числа меньшие A: {String.*Join*(" ", MinA)}");  
Console.WriteLine($"Числа меньшие B: {String.Join(" ", MoreB)}");



  
public class StudentsInfo : IComparer  
{  
 public int Compare(object x, object y)  
 {  
 if (object.*Equals*(x, y)) return 0;  
 if (x.Equals(null)) return -1;  
 if (y.Equals(null)) return 1;  
 return ((Student)x).Number.CompareTo(((Student)y).Number);  
 }  
}  
  
public struct Student  
{  
 public string Fio { get; set; }  
 public int Number { get; set; }  
 public int[] Results;  
  
 public int[] ResExams  
 {  
 get => Results;  
 set  
 {  
 if (value.Length != 3) throw new ArgumentException(nameof(ResExams));  
 Results = value.Take(3).ToArray();  
 }  
 }  
  
 public Student(string fio, int number, string[] results)  
 {  
 Fio = fio;  
 Number = number;  
 try  
 {  
 Results = results.Select(x => int.*Parse*(x)).ToArray();  
 }  
 catch (Exception e)  
 {  
 Console.*WriteLine*(e);  
 throw;  
 }  
 }  
  
 public bool IsDone  
 {  
 get  
 {  
 if (Results.Count(x => x < 3) != 0)  
 return false;  
 return true;  
 }  
 }  
  
 public override string ToString()  
 {  
 return $"ФИО: {Fio}, Номер группы: {Number}, Результаты экзаменов: {String.*Join*(", ", Results)};";  
 }  
}

string[] st = File.*ReadAllLines*("Task3.txt");  
ArrayList students = new ArrayList(st.Length);  
for (int i = 0; i < st.Length; i++)  
{  
 string line = st[i];  
 string[] f = line.Split(';', StringSplitOptions.*RemoveEmptyEntries*);  
 Student student = new Student(f[0], Convert.*ToInt32*(f[1]),  
 f[2].Split(',', StringSplitOptions.*RemoveEmptyEntries*).ToArray());  
 students.Add(student);  
}  
  
  
students.Sort(new StudentsInfo());  
foreach (var it in students)  
 Console.*WriteLine*(it);  
Console.*WriteLine*();  
if (File.*Exists*("Task3.txt"))  
 File.*Create*("Task3.txt").Close();  
using (var writer = new StreamWriter("Task3.txt"))  
{  
 var tmp = students.Cast<Student>().Where(X => X.IsDone).ToArray();  
 writer.Write(String.*Join*(";\n", tmp));  
}



public class HashMusicCatalog  
{  
 Dictionary<string, MusicCatalog> \_catalogMusic;  
  
 public HashMusicCatalog(string[] nameDisks, MusicCatalog[] titleMusic)  
 {  
 if (nameDisks.Length != titleMusic.Length &&  
 nameDisks.Contains(null) == false && titleMusic.Contains(null) == false)  
 throw new ArgumentException("Error");  
  
 \_catalogMusic = new Dictionary<string, MusicCatalog>();  
 for (int i = 0; i < nameDisks.Length; i++)  
 {  
 \_catalogMusic.Add(nameDisks[i], titleMusic[i]);  
 }  
 }  
  
 public HashMusicCatalog(Dictionary<string, MusicCatalog>? catalogMusic)  
 {  
 \_catalogMusic = catalogMusic ?? new Dictionary<string, MusicCatalog>();  
 }  
  
  
 public void AddDisk(string nameDisk, MusicCatalog diskMusic)  
 {  
 if (!\_catalogMusic.Keys.Contains(nameDisk))  
 {  
 \_catalogMusic.Add(nameDisk, diskMusic);  
 }  
 else Console.*WriteLine*("Error");  
 }  
  
 public void RemoveDisk(string nameDisk)  
 {  
 if (\_catalogMusic.Keys.Contains(nameDisk))  
 {  
 \_catalogMusic.Remove(nameDisk);  
 }  
 else Console.*WriteLine*("Error");  
 }  
  
  
 public void AddMusic(string nameDisk, string nameMusic, string nameAuthor = "Unknown")  
 {  
 if (\_catalogMusic.Keys.Contains(nameDisk))  
 {  
 Console.*WriteLine*("Error");  
 \_catalogMusic[nameDisk].Add((nameMusic, nameAuthor));  
 }  
 else Console.*WriteLine*("Error");  
 }  
  
 public void RemoveMusic(string nameDisk, string nameMusic)  
 {  
 if (\_catalogMusic.Keys.Contains(nameDisk))  
 {  
 \_catalogMusic[nameDisk].Remove(nameMusic);  
 }  
 else Console.*WriteLine*("Error");  
 }  
  
  
 public void ShowAllMusicFromCatalogMusic()  
 {  
 foreach (string nameDisk in \_catalogMusic.Keys)  
 {  
 Console.*WriteLine*($"Disk \"{nameDisk}\": ");  
 Console.*WriteLine*(\_catalogMusic[nameDisk].ToString());  
 }  
 }  
  
 public void ShowAllMusicFromDisk(string nameDisk)  
 {  
 if (\_catalogMusic.ContainsKey(nameDisk))  
 {  
 Console.*WriteLine*(\_catalogMusic[nameDisk]);  
 }  
 }  
  
 public void ShowAllMusicOfAuthor(string nameAuthor)  
 {  
 Console.*WriteLine*($"From {nameAuthor}: ");  
 foreach (var key in \_catalogMusic.Keys)  
 {  
 Console.*WriteLine*(String.*Join*(", ", \_catalogMusic[key].GetMusicOfAuthor(nameAuthor)));  
 }  
 }  
}

public class MusicCatalog  
{  
 private List<ValueTuple<string, string>> \_music;  
  
 public MusicCatalog(string[] music, string[] authors)  
 {  
 if (music.Length != authors.Length)  
 throw new ArgumentException("Error");  
 \_music = new List<(string, string)>();  
 for (int i = 0; i < music.Length; i++)  
 {  
 \_music.Add((music[i], authors[i]));  
 }  
 }  
  
 public MusicCatalog() => \_music = new List<ValueTuple<string, string>>();  
  
 public void Add(ValueTuple<string, string> newMusic)  
 {  
 if (!\_music.Contains(newMusic)) \_music.Add(newMusic);  
 }  
  
 public void Remove(ValueTuple<string, string> newMusic)  
 {  
 if (\_music.Contains(newMusic)) \_music.Remove(newMusic);  
 }  
  
 public void Add(string nameMusic, string nameAuthor)  
 {  
 if (!\_music.Contains((nameMusic, nameAuthor))) \_music.Add((nameMusic, nameAuthor));  
 }  
  
 public void Remove(string nameMusic, string nameAuthor)  
 {  
 if (\_music.Contains((nameMusic, nameAuthor))) \_music.Remove((nameMusic, nameAuthor));  
 }  
  
 public void Add(string nameMusic)  
 {  
 if (!\_music.Contains((nameMusic, "Unknown")))  
 \_music.Add((nameMusic, "Unknown"));  
 }  
  
 public void Remove(string nameMusic)  
 {  
 \_music.Remove(\_music.FirstOrDefault(x => x.Item1 == nameMusic));  
 }  
  
 public (string, string) this[string nameMusic]  
 {  
 get { return \_music.FirstOrDefault(x => x.Item1 == nameMusic); }  
 set  
 {  
 for (int i = 0; i < \_music.Count; i++)  
 {  
 if (nameMusic == \_music[i].Item1)  
 {  
 \_music[i] = value;  
 break;  
 }  
 }  
 }  
 }  
  
 public string[] GetMusicOfAuthor(string nameAuthor)  
 {  
 string[] musicOfAuthor = \_music.Where(x => x.Item2 == nameAuthor).Select(x => x.Item1).ToArray();  
 return musicOfAuthor;  
 }  
  
 public override string ToString()  
 {  
 return  
 $"Musics: {String.*Join*(", ", \_music.Select(x => x.Item1).ToArray())}.\n" +  
 $"Authors: {String.*Join*(", ", \_music.Select(x => x.Item2).ToArray())}.";  
 }  
}

Random rand = new Random();  
string[] music = { "Там где нас нет", "Детектор лжи", "То густо то пусто" };  
string[] author = { "Oxxxymiron", "Egor Kreed", "Vlad" };  
string[] disk = { "1", "2", "3" };  
  
MusicCatalog t1 = new MusicCatalog(music, author);  
MusicCatalog t2 = new MusicCatalog(music.Select(x => $"{x} {rand.Next(0,10)}").ToArray(), author);  
MusicCatalog t3 = new MusicCatalog(music.Select(x => $"{x} {rand.Next(0,10)}").ToArray(), author);  
HashMusicCatalog hash = new HashMusicCatalog(disk, new MusicCatalog[] { t1, t2, t3});  
hash.ShowAllMusicFromCatalogMusic();  
hash.ShowAllMusicFromDisk("1");  
hash.RemoveMusic("1","Там где нас нет");  
hash.ShowAllMusicOfAuthor("Oxxxymiron");

