Homework: Functional Programming

This document defines the homework assignments from the "OOP" Course @ Software University. Please submit as homework a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems. The solutions should be written in C#.

Problem 1. StringBuilder Extensions

Implement the following **extension methods** for the class **StringBuilder**:

- **Substring(int startIndex, int length)** – returns a new **String** object, containing the elements in the given range. Throw an exception when the range is invalid.
- **RemoveText(string text)** – removes all occurrences of the specified text (case-insensitive) from the **StringBuilder**. The method should not create a new **StringBuilder**, but should modify the existing one and return it as a result.
- **AppendAll(IEnumerable<T> items)** – appends the string representations of all items from the specified collection. Use **ToString()** to convert from **T** to **string**.

Write a program to demonstrate that your new extension methods work correctly.

Problem 2. Custom LINQ Extension Methods

Create your own LINQ extension methods:

- **public static IEnumerable<T> WhereNot<T>(this IEnumerable<T> collection, Func<T, bool> predicate) { … }** – works just like **Where(predicate)** but filters the non-matching items from the collection.
- **public static IEnumerable<T> Repeat<T>(this IEnumerable<T> collection, int count) { … }** – repeats the collection count **times**.
- **public static IEnumerable<string> WhereEndsWith<string>(this IEnumerable<string> collection, this IEnumerable<string> suffixes) { … }** – filters all items from the collection that ends with some of the specified **suffixes**.

Problem 3. Class Student

Create a class **Student** with properties **FirstName**, **LastName**, **FacultyNumber**, **Phone**, **Email**, **Marks** (**IList<int>**), **GroupNumber**. Create a **List<Student>** with sample students. These students will be used for the next few tasks.

Problem 4. Students by Group

Print all students from group number 2. Use LINQ query. Order the students by **FirstName**.

Problem 5. Students by First and Last Name

Print all students whose first name is before its last name alphabetically. Use LINQ query.

Problem 6. Students by Age

Write a LINQ query that finds the first name and last name of all students with age between 18 and 24.

Problem 7. Sort Students

Using the extension methods **OrderBy()** and **ThenBy()** with lambda expressions sort the students by first name and last name in descending order. Rewrite the same with LINQ.

Problem 8. Filter Students by Email Domain

Print all students that have email **@abv.bg**. Use LINQ.

Problem 9. Filter Students by Phone

Print all students with phones in Sofia (starting with 02 / +3592 / +359 2). Use LINQ.

Problem 10. Excellent Students

Print all students that have **at least one mark Excellent (6)**. Using LINQ first select them into a new anonymous class that holds **{ FullName + Marks}**.

Problem 11. Weak Students

Write a similar program to the previous one to extract the **students with exactly two marks "2"**. Use extension methods.

Problem 12. Students Enrolled in 2014

Extract and print the **Marks** of the students that **enrolled in 2014** (the students from 2014 have 14 as their 5-th and 6-th digit in the **FacultyNumber**).

Problem 13. * Students by Groups

Write a program that extracts and prints all students **grouped by GroupName** and then prints them to the console. Print all group names along with the students in each group. Use the "**group by into**" LINQ operator.

Problem 14. * Students Joined To Specialties

Create a class **StudentSpecialty** that holds **SpecialtyName** and **FacultyNumber**. Create a list of **StudentSpecialty** that specifies for eachs student his specialty. Print all students along with their faculty number and specialty. Use the "**join**" LINQ operator.

Problem 15. ** LINQ to Excel

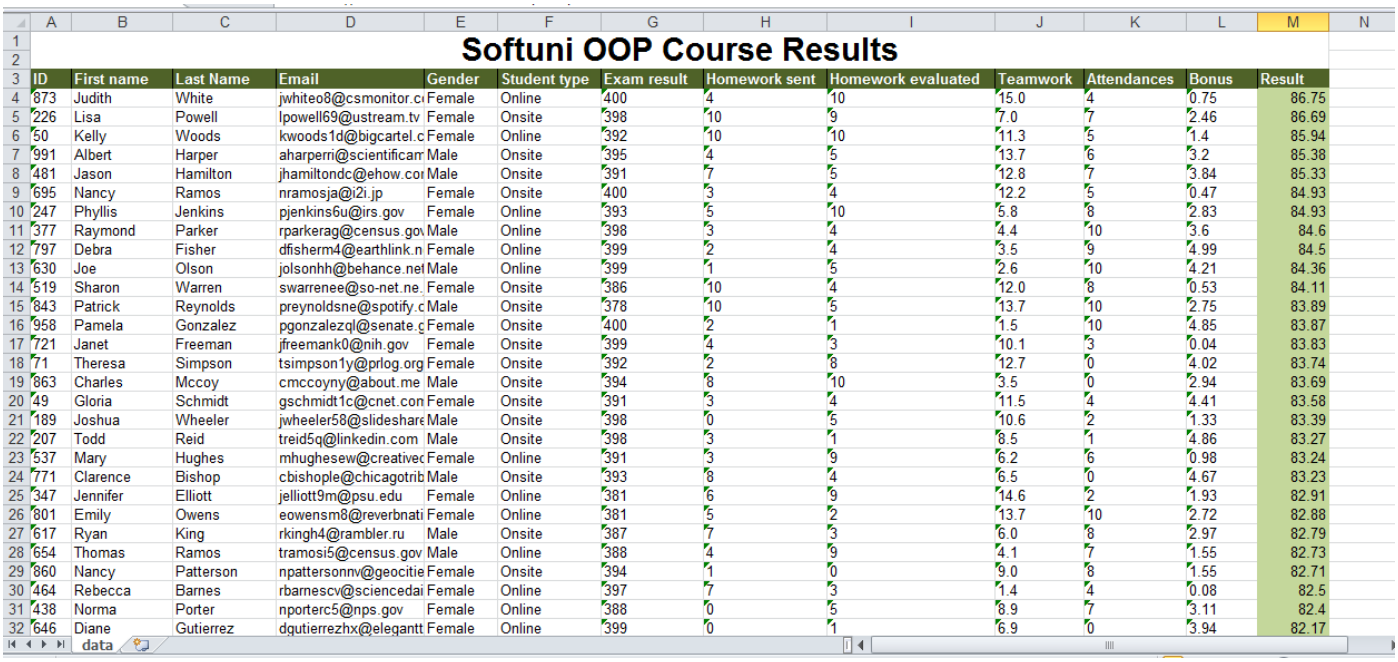
Write a C# program to create an Excel file like the one below using an external library such as LinqToExcel.

You are given as **input** course data about **1000 students** in a **.txt** file (space-separated values). Each line in the input holds **ID**, **first name**, **last name**, **email**, **gender**, **student type**, **exam result**, **homework sent**, **homework evaluated**, **teamwork score**, **attendances count**, **bonus**, **result**.

As **result** generate and open the Excel file.

- Create a class **Student** that holds all the necessary data fields from the file. Add a method **CalculateResult()** that calculates the total course **result** of a student using the formula ***(exam result + homework sent + homework evaluated + teamwork + attendances + bonus) / 5***.

- Create a **Student** object for each student from the file and store it in some collection. **Filter** only the **online students** and sort them by their **course result**. Print the result as an Excel table. Styling the table is not required.



Softuni OOP Course Results

ID	First name	Last Name	Email	Gender	Student type	Exam result	Homework sent	Homework evaluated	Teamwork	Attendances	Bonus	Result
873	Judith	White	jwhiteo8@csmonitor.c	Female	Online	400	4	10	15.0	4	0.75	86.75
226	Lisa	Powell	lpowell69@ustream.tv	Female	Onsite	398	10	9	7.0	7	2.46	86.69
50	Kelly	Woods	kwoods1d@bigcartel.c	Female	Online	392	10	10	11.3	5	1.4	85.94
991	Albert	Harper	aharperri@scientificam	Male	Onsite	395	4	5	13.7	6	3.2	85.38
481	Jason	Hamilton	jhamiltondc@ehow.co	Male	Onsite	391	7	5	12.8	7	3.84	85.33
695	Nancy	Ramos	nramosja@i2i.jp	Female	Onsite	400	3	4	12.2	5	0.47	84.93
247	Phyllis	Jenkins	pjenkins6u@irs.gov	Female	Online	393	5	10	5.8	8	2.83	84.93
377	Raymond	Parker	rparkerag@census.gov	Male	Online	398	3	4	4.4	10	3.6	84.6
797	Debra	Fisher	dfisherm4@earthlink.n	Female	Online	399	2	4	3.5	9	4.99	84.5
630	Joe	Olson	jolsonhh@behance.net	Male	Online	399	1	5	2.6	10	4.21	84.36
519	Sharon	Warren	swarrenee@so-net.ne.	Female	Onsite	386	10	4	12.0	8	0.53	84.11
843	Patrick	Reynolds	preynoldsne@spotify.c	Male	Onsite	378	10	5	13.7	10	2.75	83.89
958	Pamela	Gonzalez	pgonzalezql@senate.g	Female	Onsite	400	2	1	1.5	10	4.85	83.87
721	Janet	Freeman	jfreemank0@nih.gov	Female	Onsite	399	4	3	10.1	3	0.04	83.83
71	Theresa	Simpson	tsimpson1y@prlog.org	Female	Onsite	392	2	8	12.7	0	4.02	83.74
863	Charles	Mccoy	cmccoyny@about.me	Male	Onsite	394	8	10	3.5	0	2.94	83.69
49	Gloria	Schmidt	gschmidt1c@cnet.con	Female	Onsite	391	3	4	11.5	4	4.41	83.58
189	Joshua	Wheeler	jwheeler58@slideshare	Male	Onsite	398	0	5	10.6	2	1.33	83.39
207	Todd	Reid	treid5q@linkedin.com	Male	Onsite	398	3	1	8.5	1	4.86	83.27
537	Mary	Hughes	mhughesew@creativec	Female	Online	391	3	9	6.2	6	0.98	83.24
771	Clarence	Bishop	cbishople@chicagotrib	Male	Onsite	393	8	4	6.5	0	4.67	83.23
347	Jennifer	Elliott	jelliott9m@psu.edu	Female	Online	381	6	9	14.6	2	1.93	82.91
801	Emily	Owens	eowensm8@reverbnati	Female	Online	381	5	2	13.7	10	2.72	82.88
617	Ryan	King	rkingh4@rambler.ru	Male	Onsite	387	7	3	6.0	8	2.97	82.79
654	Thomas	Ramos	tramosi5@census.gov	Male	Online	388	4	9	4.1	7	1.55	82.73
860	Nancy	Patterson	npattersonnv@geocitie	Female	Onsite	394	1	0	9.0	8	1.55	82.71
464	Rebecca	Barnes	rbarnescv@sciencedai	Female	Online	397	7	3	1.4	4	0.08	82.5
438	Norma	Porter	nporterc5@nps.gov	Female	Online	388	0	5	8.9	7	3.11	82.4
646	Diane	Gutierrez	dgutierrezhx@elegantt	Female	Online	399	0	1	6.9	0	3.94	82.17