# Exercise: Functional Programming

This document defines in-class exercise problems from the [“Advanced C#“ Course @ Software University](http://softuni.bg/courses/advanced-csharp/). You are presented with some problems and certain steps you need to take in order to accomplish the tasks.

## Problem 2. Filter Students

You have a colleague who’s just started studying for the OOP course in SoftUni. He has created a class Student and he wants to extract some student data from a file and filter it. He’s pretty good in terms of object oriented programming, but he doesn’t know how to work with files and he has no idea how to use LINQ queries to achieve his goals.

You’re given a solution called FunctionalProgrammingExercise.sln containing a single project (LinqExercise) and a .txt file with data for 1000 students. The Student class is implemented and your colleague has helpfully written for you the boring splitting and parsing of lines from the file. He takes the data from each line and parses it to the correct format, creates a new Student object and adds it to a list.

So, the hard parts are already done. Finish the task by following the steps below.

### Step 1. Read the Data

Use a **StreamReader** to take one line at a time from the Students-data.txt file. Remember to stop when the file ends and make sure you don’t try to parse the first line (it’s a header line with no actual data). The rest of the logic behind filling the list is done.

### Step 2. Execute Queries on the Data

Now that you have a collection to work with, you can start filtering and manipulating it. Note that by default if you try to print a Student object on the console you’ll get only their name and whether they are online or on-site. To make sure your queries are correct, when printing the students, print also the relevant data (e.g. if you want to sort by score, print the score as well).

**Filtering and ordering:**

* Extract all male students and print them on the console.

|  |
| --- |
| **Sample Output** |
| Bruce Arnold - Male  Jeremy Chavez - Male  Douglas Kelly - Male  Alan Ellis - Male  Terry Hill – Male  Johnny Owens - Male  Billy Bowman - Male  Gregory Kim – Male  Howard Fernandez - Male  Carlos Fuller - Male  Harry Nelson - Male  … |

* Extract all students whose first names start with the letter 'A' and print them on the console.

|  |
| --- |
| **Sample Output** |
| Aaron Gibson  Antonio Gonzalez  Andrea Harper  Alice Baker  Anne Freeman  … |

* Extract all online students with exam score greater than or equal to 350 and print them on the console.

|  |
| --- |
| **Sample Output** |
| Lois Patterson (Online) - 358  Kelly Woods (Online) - 392  Roger Adams (Online) - 375  James Nelson (Online) - 380  George Stanley (Online) – 386  … |

* Extract all online students with score >= 300, order them by exam score in descending order and print them on the console.

|  |
| --- |
| **Sample Output** |
| Judith White (Online) - 400  Joe Olson (Online) - 399  Diane Gutierrez (Online) - 399  Debra Fisher (Online) - 399  Raymond Parker (Online) - 398  Rebecca Barnes (Online) - 397  Phyllis Jenkins (Online) – 393  … |

* Extract all students without any homework sent, order them by first name, then by last name and print them on the console.

|  |
| --- |
| **Sample Output** |
| Alan Shaw (0 homeworks sent)  Albert Patterson (0 homeworks sent)  Alice Morrison (0 homeworks sent)  Antonio Gutierrez (0 homeworks sent)  Arthur Hayes (0 homeworks sent)  Betty Matthews (0 homeworks sent)  … |

**Transforming:**

So far your queries took all the data from a student. It’s usually preferable to extract only the data that you actually want. Use the **Select()** method and transform the data to retrieve only the necessary parts in the following queries (use anonymous types when selecting more than one property):

* Extract the emails of all on-site students.

|  |
| --- |
| **Sample Output** |
| bgardnerjy@angelfire.com  jfreemank0@nih.gov  dfoxk4@altervista.org  pnguyenk5@salon.com  jhudsonk6@mail.ru  sgibsonk8@uiuc.edu  … |

* Extract the exam results and attendance counts of all on-site students who have less than 5 attendances.

|  |
| --- |
| **Sample Output** |
| Result: 204, attendances: 2  Result: 233, attendances: 1  Result: 175, attendances: 2  Result: 230, attendances: 3  Result: 72, attendances: 1  Result: 368, attendances: 1  Result: 100, attendances: 0  … |

**Aggregating:**

* Find the number of students who have a bonus of 4 or more.

|  |
| --- |
| **Sample Output** |
| 193 |

* Find the average exam score of online and on-site students (don’t try doing it in a single query, just make two☺).

|  |
| --- |
| **Sample Output** |
| Online average: 197.576446280992  Onsite average: 192.875968992248 |

* Find the number of students who have a teamwork score equal to the maximal teamwork score.

|  |
| --- |
| **Sample Output** |
| 7 |

**Grouping:**

* Group the students by the initials of their first names; sort the groups alphabetically and print them like the example below.

|  |
| --- |
| **Sample Output** |
| A  Aaron Gibson  Antonio Gonzalez  Andrea Harper  …  B  Billy Romero  Bonnie Shaw  Beverly Clark  Billy Bowman  … |

* Group the students by type (online/on-site) and sort them by exam score in descending order.

|  |
| --- |
| **Sample Output (partial data)** |
| Online  Kelly Woods - 392  Lois Patterson - 358  Heather Knight - 346  Cheryl Gray – 315  …  Onsite  Gloria Schmidt - 391  Lillian Hart - 337  Jeffrey Gibson - 316  Susan Boyd – 310  … |