Programming II (420-2P6-AB)  
Winter 2021

**Project [Worth 20%]**

**Part 1 (5%) due: Monday, May 17th, 2021 at 23:59 on Lea  
Part 2 (15%) due: Sunday, May 30th, 2021 at 23:59 on Lea**

**No late submissions accepted.**

## Objective

The objective of this assignment involves reading data from 2 text files. Data will be processed as objects and the program will generate 4 reports which will be written to 4 different text files.

**Important!** Use this exact namespace: **Project2P6.**

Store all text files in the same place as the .EXE: ProjectName\bin\Debug folder

## Tasks

Data is saved in 2 CSV files: employees.txt and employeesHours.txt. Read the data from the files using streams, as previously seen.   
  
The first, named employees.txt, is a list of employees; here is a sample file:

2745,Brown,Jane,15.50

1121,Smith,Brian,17.75

8343,Jones,Nora,15.00

4231,White,Allan,19.00

5566,Moss,Peter,13.45

2002,Clark,Mary,14.60

9223,Edwards,Kim,18.00

2341,Young,Ann,21.88

5342,Peters,Peter,16.00

1885,Hugo,Boss,65.50

1964,Valentina,Tereshkova,77.77

* The file has 4 fields: *employee number*, *last name*, *first name* and *hourly rate*.
* The fields are separated by a comma: ‘,’.

**Notes:** Assume the file has no errors, and there is at least 1 employee.

The second file, named employeesHours.txt, contains the amount of time an employee worked, each time they worked. Here is a sample:

2745,05:23:00

5566,01:00:00

8343,04:15:00

1885,04:20:00

2002,03:00:00

1964,10:49:00

1121,08:40:00

5342,16:05:00

5342,15:15:00

4231,09:29:00

2745,07:50:00

9223,06:00:00

8343,09:10:00

1885,11:10:00

1121,07:49:00

4231,04:27:00

2002,08:58:00

5342,04:22:00

1964,22:22:00

2341,02:11:00

1885,09:59:00

1121,01:45:00

2745,00:30:00

8343,01:20:00

* This file has 2 fields: *employee* *number* and a *timestamp*, which represents how long they worked.

**Notes:**   
A single employee can work on several occasions.   
Every employee number in this file, appears in the employee file above.   
The program, given the above files, will produce the following 4 reports: (as shown)

**employeeNumberOrder.txt**

Emp # Last Name First Name Time Worked Hourly Wage Pay

----- --------------- --------------- ----------- ----------- ------------

1121 Smith Brian 18:14:00 $17.75 $323.64

1885 Hugo Boss 25:29:00 $65.50 $1,669.16

1964 Valentina Tereshkova 33:11:00 $77.77 $2,580.67

2002 Clark Mary 11:58:00 $14.60 $174.71

2341 Young Ann 02:11:00 $21.88 $47.77

2745 Brown Jane 13:43:00 $15.50 $212.61

4231 White Allan 13:56:00 $19.00 $264.73

5342 Peters Peter 35:42:00 $16.00 $571.20

5566 Moss Peter 01:00:00 $13.45 $13.45

8343 Jones Nora 14:45:00 $15.00 $221.25

9223 Edwards Kim 06:00:00 $18.00 $108.00

Total time worked: 176:09:00  
Total pay: $6,187.19

**nameOrder.txt**

Emp # Last Name First Name Time Worked Hourly Wage Pay

----- --------------- --------------- ----------- ----------- ------------

2745 Brown Jane 13:43:00 $15.50 $212.61

2002 Clark Mary 11:58:00 $14.60 $174.71

9223 Edwards Kim 06:00:00 $18.00 $108.00

1885 Hugo Boss 25:29:00 $65.50 $1,669.16

8343 Jones Nora 14:45:00 $15.00 $221.25

5566 Moss Peter 01:00:00 $13.45 $13.45

5342 Peters Peter 35:42:00 $16.00 $571.20

1121 Smith Brian 18:14:00 $17.75 $323.64

1964 Valentina Tereshkova 33:11:00 $77.77 $2,580.67

4231 White Allan 13:56:00 $19.00 $264.73

2341 Young Ann 02:11:00 $21.88 $47.77

Total time worked: 176:09:00

Total pay: $6,187.19

**timeOrder.txt**

Emp # Last Name First Name Time Worked Hourly Wage Pay

----- --------------- --------------- ----------- ----------- ------------

5342 Peters Peter 35:42:00 $16.00 $571.20

1964 Valentina Tereshkova 33:11:00 $77.77 $2,580.67

1885 Hugo Boss 25:29:00 $65.50 $1,669.16

1121 Smith Brian 18:14:00 $17.75 $323.64

8343 Jones Nora 14:45:00 $15.00 $221.25

4231 White Allan 13:56:00 $19.00 $264.73

2745 Brown Jane 13:43:00 $15.50 $212.61

2002 Clark Mary 11:58:00 $14.60 $174.71

9223 Edwards Kim 06:00:00 $18.00 $108.00

2341 Young Ann 02:11:00 $21.88 $47.77

5566 Moss Peter 01:00:00 $13.45 $13.45

Total time worked: 176:09:00

Total pay: $6,187.19

**payOrder.txt**

Emp # Last Name First Name Time Worked Hourly Wage Pay

----- --------------- --------------- ----------- ----------- ------------

5566 Moss Peter 01:00:00 $13.45 $13.45

2341 Young Ann 02:11:00 $21.88 $47.77

9223 Edwards Kim 06:00:00 $18.00 $108.00

2002 Clark Mary 11:58:00 $14.60 $174.71

2745 Brown Jane 13:43:00 $15.50 $212.61

8343 Jones Nora 14:45:00 $15.00 $221.25

4231 White Allan 13:56:00 $19.00 $264.73

1121 Smith Brian 18:14:00 $17.75 $323.64

5342 Peters Peter 35:42:00 $16.00 $571.20

1885 Hugo Boss 25:29:00 $65.50 $1,669.16

1964 Valentina Tereshkova 33:11:00 $77.77 $2,580.67

Total time worked: 176:09:00

Total pay: $6,187.19

The 4 reports contain the **same data**. The only difference is the order in which the records are printed out. Each report should be written to a different text file, with the given name above.

Every time you write to a report file, also **output to the screen**.

**You will need all the following:**

1. Your “TimeStamp” Class from Assignment 4. It should be modified so that a TimeStamp can have more than 23 hours. I.E. an Employee can work 35 hours in a week.
2. A new Class named “Employee” to keep track of each Employee.
   1. Add data and methods members to make your code more useful.
   2. Add constructors.
   3. Use *Properties.*
3. Use a List only. DO NOT USE arrays.
4. Avoid magic numbers in your code. (Use constants or variables)
5. You will need to implement each of the following methods within your “Program.cs” source file:

**// Populates a list of employees from the provided files. Return true if reading from file worked, false otherwise.**public static bool GenerateEmployeeListFromFile(List<Employee> employeeList, string fileName)  
  
**// Adds working hours stated in the provided file to the employees. Return true if reading from file worked, false otherwise.**public static bool ProcessTimeWorkedFile(List<Employee> employeeList, string fileName)  
  
**//Generated a text report of the provided employee list to the provided file. Return true if writing to file worked, false otherwise.**public static boolPrintReport(List<Employee> employeeList, string fileName);

**//Given an employee number and hours worked, update the specific employee in the list. Return true if employeeNumber is found in the list, false otherwise.**public static bool AddTimeWorkedToEmployee(List<Employee> employeeList, int employeeNumber, TimeStamp timeWorked);

**//Sorting Methods to change the order of elements in the list.  
//HINT: Use LINQ to sort the data and return a sorted list.**public static List<Employee> SortEmployeeListByLastName(List<Employee> employeeList)  
public static List<Employee> SortEmployeeListByTimeWorkedDesc(List<Employee> employeeList)  
public static List<Employee> SortEmployeeListByPay(List<Employee> employeeList)

***Note:*** You will be learning how to sort data using LINQ in class.

You may use the provided data files for testing, but you must also make your own data files, which you must hand-in. Your data files must have more records than the provided one.

LINQ tutorials:

* [Standard Query Operators (tutorialsteacher.com)](https://www.tutorialsteacher.com/linq/linq-standard-query-operators)
* <https://linqsamples.com/tutorials/linq-for-beginners>
* <https://linqsamples.com/linq-to-objects/ordering/OrderBy-numbers-linq>
* <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>
* <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/data-transformations-with-linq>

## Project Parts

### Part 1:

**Plan the program (5% of final grade). Due: Monday, May 17th, 2021 at 23:59.**No late submissions accepted and 0/5 will be given if late.

Create a **detailed** document that includes your **detailed** plan which includes outline and pseudo-code for:

* 1. Design the Employee class. Outline the backing fields, properties, constructors, methods.
  2. Functional decomposition design: before implementing the methods, break down the program into methods and have enough of them. Write the method headers like the teacher provided you in assignment 4: comments and /// documentation. For each method explain:
     1. Inputs.
     2. Outputs.
     3. Task description in details, step by step.
  3. Main()
  4. DO NOT CODE FIRST AND THEN CREATE THE PLAN. THAT’S AN AUTOMATIC 0/5. CLAUDIU WILL KNOW!

### Part 2:

**Implement the plan (15% of final grade). Due May 30th, 2021 at 23:59.** No late submissions accepted and 0% will be given if late.

**Important!** Use this exact namespace: **Project2P6.**

Store all text files in the same place as the .EXE: ProjectName\bin\Debug folder

# Submission:

1. Add program header and comment your code and add /// summary : worth 5%.
2. Include the tester’s name in the program’s header: worth 5%.
3. **For all requirement, the class throws an exception when invalid data is inputted in its fields.**
4. All programs should have pause at the very end: Console.ReadKey();
5. In one folder place the entire Visual Studio folder, including the .SLN and .CS files.
6. Compress this entire folder into a ZIP file (NO other format) and submit this on LEA.