Lab: Streams, Files and Directories

Problems for the "C# Advanced" course @ SoftUni

You can check your solutions in Judge

NOTE: For these problems follow the instructions for the required methods and classes. For each problem **submit zipped folder** of your project **without** the **"bin"** and **"obj"** folders in it.

I. File Operations

1. Odd Lines

Write a program that reads a text file (e. g. **`input.txt`**) and writes every **odd** line in another file. Line numbers **start from 0**.

Note: use the following structure:

```
namespace OddLines
{
    public class OddLines
    {
        static void Main()
        {
            string inputFilePath = @"..\..\..\Files\input.txt";
            string outputFilePath = @"..\..\..\Files\output.txt";

            ExtractOddLines(inputFilePath, outputFilePath);
        }

        public static void ExtractOddLines(string inputFilePath, string
outputFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

input.txt	output.txt

`Two households, both alike in dignity,`
`In fair Verona, where we lay our scene,`
`From ancient grudge break to new mutiny,`
`Where civil blood makes civil hands unclean.`
`From forth the fatal loins of these two foes`
`A pair of star-cross'd lovers take their life;`
`Whose misadventured piteous overthrows`
`Do with their death bury their parents' strife.`
	`In fair Verona, where we lay our scene,`
	`Where civil blood makes civil hands unclean.`
	`A pair of star-cross'd lovers take their life;`
	`Do with their death bury their parents' strife`

2. Line Numbers

Write a program that **reads a text file** (e. g. **`input.txt`**) and **inserts line numbers** in front of each of its lines. The result should be **written to another text file** (e. g. **`output.txt`**). Use **`StreamReader`** and **`StreamWriter`**.

NOTE: use the following structure:

```
namespace LineNumbers
{
    public class LineNumbers
    {
        static void Main()
        {
            string inputPath = @"..\..\..\Files\input.txt";
            string outputPath = @"..\..\..\Files\output.txt";

            RewriteFileWithLineNumbers(inputPath, outputPath);
        }

        public static void RewriteFileWithLineNumbers(string inputFilePath, string
outputFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

input.txt	output.txt
Two households, both alike in dignity, In fair Verona, where we lay our scene, From ancient grudge break to new mutiny, Where civil blood makes civil hands unclean. From forth the fatal loins of these two foes A pair of star-cross'd lovers take their life; Whose misadventured piteous overthrows Do with their death bury their parents' strife.	1. Two households, both alike in dignity, 2. In fair Verona, where we lay our scene, 3. From ancient grudge break to new mutiny, 4. Where civil blood makes civil hands unclean. 5. From forth the fatal loins of these two foes 6. A pair of star-cross'd lovers take their life; 7. Whose misadventured piteous overthrows 8. Do with their death bury their parents' strife.

3. Word Count

Write a program that **reads a list of words** from a given file (e. g. **`words.txt`**) and finds how many times each of the words occurs in another file (e. g. **`text.txt`**). Matching should be **case-insensitive**. The **result** should be written to an output text file (e. g. **`output.txt`**). Sort the words by frequency in descending order.

NOTE: use the following structure:

```
namespace WordCount
{
    public class WordCount
    {
        static void Main()
        {
            string wordPath = @"..\..\..\Files\words.txt";
            string textPath = @"..\..\..\Files\text.txt";
            string outputPath = @"..\..\..\Files\output.txt";

            CalculateWordCounts(wordPath, textPath, outputPath);
        }

        public static void CalculateWordCounts(string wordsFilePath, string
textFilePath, string outputFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

words.txt	text.txt	output.txt

`quick is fault`	`-I was quick to judge him, but it wasn't his fault.`
	`-Is this some kind of joke?! Is it?`
	`-Quick, hide here…It is safer.`
		`is - 3`
		`quick - 2`
		`fault - 1`

4. Merge Files

Write a program that reads the contents of **two input text files** (e. g. **`input1.txt`** and **`input2.txt`**) and **merges them line by line** together into a third text file (e. g. **`output.txt`**). The merging is done as follows:

- Line 1 from input1.txt
- Line 1 from input2.txt
- Line 2 from input1.txt
- Line 2 from input2.txt
- …

If some of the files have more lines than the other, append at the end of the output the lines, which cannot be matched with the other file.

NOTE: use the following structure:

```
namespace MergeFiles
{
    public class MergeFiles
    {
        static void Main()
        {
            var firstInputFilePath = @"..\..\..\Files\input1.txt";
            var secondInputFilePath = @"..\..\..\Files\input2.txt";
            var outputFilePath = @"..\..\..\Files\output.txt";

            MergeTextFiles(firstInputFilePath, secondInputFilePath,
outputFilePath);
```

```
        }

        public static void MergeTextFiles(string firstInputFilePath, string
secondInputFilePath, string outputFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

input1.txt	input2.txt	output.txt
1 3 5	2 4 6 7	1 2 3 4 5 6 7

II. Directory Operations

5. Extract Special Bytes

You are given a binary file (e. g. **`example.png`**) and a text file (e. g. **`bytes.txt`**), holding a list of bytes in the range [0…255]. Write a program to extract occurrences of all given bytes from the input file to an output binary file (e. g. **`output.bin`**).

NOTE: use the following structure:

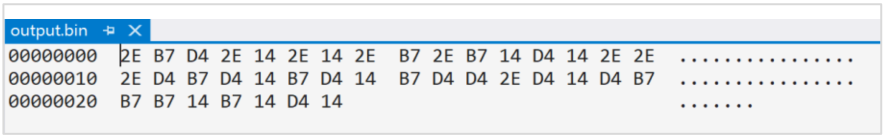
```
namespace ExtractSpecialBytes
{
    public class ExtractSpecialBytes
    {
        static void Main()
        {
            string binaryFilePath = @"..\..\..\Files\example.png";
            string bytesFilePath = @"..\..\..\Files\bytes.txt";
            string outputPath = @"..\..\..\Files\output.bin";

            ExtractBytesFromBinaryFile(binaryFilePath, bytesFilePath, outputPath);
        }

        public static void ExtractBytesFromBinaryFile(string binaryFilePath, string
bytesFilePath, string outputPath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

example.png	bytes.txt	output.bin

	20
	46
	183
	212
		
		`00000000  2E B7 D4 2E 14 2E 14 2E  B7 2E B7 14 D4 14 2E 2E  ................`
		`00000010  2E D4 B7 D4 14 B7 D4 14  B7 D4 D4 2E D4 14 D4 B7  ................`
		`00000020  B7 B7 14 B7 14 D4 14                          .......`

6. Split / Merge Binary Files

You are given an input binary file (e. g. **`example.png`**). Write a program to **split it into two equal-sized files** (e. g. **`part-1.bin`** and **`part-2.bin`**). When the input file size is an odd number, the first part should be 1 byte bigger than the second.

After splitting the input file, **join the obtained files** into a new file (e. g. **`example-joined.png`**). The obtained result file should be the same as the initial input file.

NOTE: use the following structure:

```
namespace SplitMergeBinaryFile
{
    public class SplitMergeBinaryFile
    {
        static void Main()
        {
            string sourceFilePath = @"..\..\..\Files\example.png";
            string joinedFilePath = @"..\..\..\Files\example-joined.png";
            string partOnePath = @"..\..\..\Files\part-1.bin";
            string partTwoPath = @"..\..\..\Files\part-2.bin";

            SplitBinaryFile(sourceFilePath, partOnePath, partTwoPath);
            MergeBinaryFiles(partOnePath, partTwoPath, joinedFilePath);
        }

        public static void SplitBinaryFile(string sourceFilePath, string
partOneFilePath, string partTwoFilePath)
        {
            // TODO: write your code here…
        }

        public static void MergeBinaryFiles(string partOneFilePath, string
partTwoFilePath, string joinedFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

III. Directory Operations

7. Folder Size

You are given a folder in the file system (e. g. **`TestFolder`**). Calculate the size of all files in the folder and its **subfolders.** The result should be written to another text (e. g. **`output.txt`**) file in **kilobytes**.

NOTE: use the following structure:

```
namespace FolderSize
{
    public class FolderSize
    {
        static void Main()
        {
            string folderPath = @"..\..\..\Files\TestFolder";
            string outputPath = @"..\..\..\Files\output.txt";

            GetFolderSize(folderPath, outputPath);
        }

        public static void GetFolderSize(string folderPath, string outputFilePath)
        {
            // TODO: write your code here…
        }
    }
}
```

Examples

output.txt
0.0869140625 KB