### While solving, I broke the problem in 3 subproblems:

1. How to read the input files
2. How to identify the palindrome
3. How to write to the output files

* **How to read the Input files** – There are various options. But since I have to use it only once in my program, so I preferred using *ReadAllTextAsync* method of static *File class.*
* **How to identify the palindrome** – For this I wrote a palindrome function, that takes string as input. We are comparing Ascii code from beginning and end, of each character (after converting it to lower string)
* **How to write to the input files** - There are various options. But since I have to use it only once in my program, so I preferred using *AppendAllLinesAsync* method of static *File class*.

Implementing Solid Design principles:

1. **Single Responsibility Principle** - I created separate files for each task.

* *Readwrite.cs* = for input-output tasks
* *Palindrome.cs* = for finding palindrome
* *Worker.cs* = for driving all the operations
* *Program.cs* = starting point of all the applications

1. **Dependency Inversion Principle** - In order to make each module easily testable, I followed dependency injection approach using ***AutoFac library.*** For all of my dependencies, I have used singleton Lifetime.
2. **Interface Segregation Principle** – I tried to segregate interface as much as possible, in order to avoid using unnecessary method, while implementing interface.

Input File and Output file setting:

* Input and output path can be given in *app.config* file.
* We need to give just directory name where input file. This is done to handle if there are multiple input files.
* If there are multiple input files, the output will be generated for each input files. The format of output file name will be: **inputfilename-output.extension.**

Other consideration

* We don’t need to create output directory. It will be created if we have the output path.
* If directory path keys are not present in app.config, then we will get a message on the console.
* If input directory does not exist, then we will get a message on the console.
* If input directory exists but input file is not present, then we will get a message on console.
* Read file and write file will happen only once.
* If output file is already existing then it will be overwritten.