**.Net hands-on exercice**

1. Introduction

The purpose of the current exercise is to develop an application in C# using the tools and technologies available for developers in the latest release of Microsoft .Net framework.   
The exercise tests not just coding abilities, but also architecture and software design – so sufficient thought must be given to design and test aspects of the solution.  
The solution must be efficient, flexible and robust.

1. Exercise

You are required to develop a logging server for processing and storing logging information arriving from external components. When messages are logged, all the information related to that message needs to be stored (.e.g message timestamp, textual message and whatever additional information you’d like to store).   
Eventually, the messages are stored into a single file.  
In case of a failure or stopping the logging server manually, when the logging server is started again it will continue logging from the same spot (in terms of file location, the data not yet stored will be lost)



The system is required to support a high number of logging requests concurrently with burst performance – that is the difference between the maximum and average rates of message processing are large.  
For example given a storage subsystem capable of writing 30 MB/sec reliably we can be sure that on average we will not be required to store more than 10 MB/sec of log data.   
However for short periods of time there can be bursts with data rates of 100 MB/sec and the system must be able to cope with them.  
  
An additional requirement for this server is that besides chronological ordering of log entries in the file, you are required to order entries by severity.   
For example, if 100 warning messages have been received and are being saved, and 10 error messages comes in – they will be saved before the server continues processing the rest of the warning messages. And if during processing of these 10 messages a critical message comes in, processing it will take precedence.  
The immediate effect of this, is that contrary to regular logs – the information in the file will not be strictly in chronological order, which is ok for our purposes.

Logging capabilities required are basic text logging with log level specification. Each log entry includes textual data only, no binary attachments are required.

External entities that create the log entries can be located on the same physical machine, or on another one with network connectivity, you are required to support both configurations.

Log levels are (from less severe to the more severe):

* Debug
* Info
* Warn
* Error
* Critical

1. Tasks
   1. Design the API between the external entities using the logging services and your server
   2. Develop the logging server
   3. Bonus task: develop reliable logging capability – i.e. even if the logging server is stopped and then restarted the data that was sent during downtime is not lost, but processed when the server is back up.
   4. Bonus task: develop the rolling file capability  
      Specify max. file size (N MB) in configuration and the number of log files (M) and when a file reaches size N it is renamed and a new file is started for the current logs.

So during a long run the following files are created:

* + 1. messages.log
    2. messages.log.0
    3. message.log.1
    4. …
    5. messages.log.M

Good luck!