**Development Exercise:**

Build a C&C (C2) application that has two main components:

* Server side - Manage all connected agents, by using keep alive messages, and has the ability to dispatch commands for execution to available agents and receive output results.
* Client side – Agent that connects to a command controller and waits for incoming command for execution.

Exercise Requirements:

* **Client:**
  1. The client has a predefined configuration file which will be referenced as “(configuration)” through out the document.
  2. Write a client agent that on startup connects to its server by IP:PORT (configuration), and sends keep alive messages to the server at a configurable interval.
  3. The client should listen for incoming command messages from its C&C, while the command message structure should be as follows:
     + command payload (of type: byte array)
     + command type (of type int/string),
     + command identifier (unique ID, each command must have a unique identifier) (of type: int),
     + command payload arguments (of type: string array).
  4. When a new message is received from the server. The command payload should be saved as a file (so / python / etc… dependent on the language of your choosing) in a designated directory (configuration).
  5. The client loads the payload file (mentioned in step 4) and runs a specific method (known in advance) according to a known function signature, with the received arguments from the “command payload arguments”.  
     An example for command payload file could be:
     + file upload command
     + shell execution command
     + port scanning command
     + etc.
  6. The client should transmit to the server the status of the command execution process, i.e. Received, Initialized, Running, Finished or Error.
  7. The client will transmit the command execution results in base64 encoding to the server if it was a successfully executed, or an error message if failed, with the “command identifier”.
  8. The client should manage incoming command in a queue and then execute each command in a different thread/process.
  9. The client will delete the command execution file after the result have been sent to the C&C.
* **Server:**
  1. Write a server C&C application for managing agent clients which connect to the server, and manage the liveness of each client by the keep alive messages received in the server from the clients.
  2. The server should handle a refreshing CLI (interval by configuration) which displays the C&C status and its connected clients (single field for the C&C status and a table for the client’s status).
  3. The server CLI should receive keyboard inputs from its operator for controlling its connected clients. The commands menu should be as follows:
     + Choose Operation
       - Send Command
       - Remove/Kill Client
       - Display Command Result
     + Choose Sub Menu (If needed, for example choosing command to send)
       - Single Client/Broadcast
       - Choose Client (If needed)
     + Receive Input Arguments (If needed).
     + The selection of client and command to execution should be easy to understand and operate.
  4. The server will manage the status and the information (results) about all the operations of commands that are executing by the clients (ID).  
     The operator should be able to understand the status of his clients and commands, and also be able to display the result of the commands.
  5. The server should support additional command development by plug-n-play capabilities, meaning that the CLI command choosing is controlled by iterating on a directory of prepared commands, or by configuration (should be easy to manage by the operator).

Development Requirements:

* The code should be clean and readable (easy to maintain)
* The C&C should support an easy plug-n-play feature for additional command implementation.
* The application should handle edge cases of input data and command execution of the client agents.
* The solution code should be well documented and well-structured by using classes and utilities, also write a short README file for the projects execution for the operator.
* The components (server and client) should be a Console Application projects that share common code if needed, also the CLI projects and the server and client logic should be separate projects!
* The components (server and client) should handle exceptions and logging.
* The application should be written in C++, Java or Python.
* This exercise should be completed in around a day of work.