Report for the Unit EE30241.

At submission (including the code) a report containing the description, explanation and reflection on the different exercises is required (max 3 pages per exercise, including pictures and references).

Each exercise "chapter" will have a structure as follows:

- **Introduction including a description of the task**
- **Solution explanation: which include a description of the code (only a few lines can be used when relevant for discussion) screen shots, data if any – hint: for example, in ex1 description of the protocol (commands, parser etc…)**
- **Conclusion: which includes reflection on the specific exercise and what could have been improved, what went wrong and what was good.**

Keep a logbook to help in writing the report at the end of the unit.

Exercise 1 – EE30241

This exercise is part of the marked coursework. The Report part of the Coursework will contain the reflection and description as described above.

(#1) Using C++, or Python write a program that acts as processing unit (a simple calculator).
(#2) Using a second program (C# advised if you decide to use a GUI or other similar) write a "requester" program (e.g. an interface)
These two programs will send/receive messages to request of calculations (#2) and respond with results (#1).

The communication will see the processing unit (#1) being the Server and (#2) an User Interface (e.g. GUI) as the Client receiving the user input and requesting computation from (#1).

Use sockets to establish communication and design a suitable protocol.
Ideal steps Ex1:

A) Create a server that can handle at least a client.
B) The server will function as calculator (operations min: = + - * / ^ sqrt).
C) Implement a client using C# (or Python) that can receive commands from a user.
D) Receive and send commands (you will need to create your own protocol and/or parser to decode the strings)
E) Use the GUI process to send commands to the "calculator process".
F) Send back the result and display on GUI.
G) The connection and performances need to be good and reliable, and transparency should be provided (e.g. log of communication -> saved on a txt file).

Hints:

1) You can start with stand-alone applications.
2) Build the programs as state machines will help.
3) Keep it simple the focus is on understanding the inter-processes communication **not** to make a scientific fully functioning calculator or creating a perfect user interface.