A C++ shared library able to parse a simple INI file with the following public interfaces:

- ***unsigned short load_resource(const std::string& path);***
  - allows an application to load and parse in volatile memory a INI resource file located in a standard Linux filesystem.
  - Example:

    unsigned short res = load_resource(“/tmp/example.ini”);
  - *Return values*: 0 in case of success, 1 in case of read errors, 255 in case of generic error.
- ***unsigned short get_value(const std::string& key, std::string& value);***
  - allows an application to retrieve the value of a key available in a previously loaded INI file.
  - Example:

    std::string buffer;

    unsigned short res = get_value(“section.foo.bar”, buffer);
  - *Return values*: 0 in case of success, 3 in case of missing key, 4 in case a resource file has not been loaded yet, 255 in case of generic error.
- ***unsigned short set_value(const std::string& key, const std::string &value);***
  - allows an application to store the value of a key in a previously loaded INI file. This adds or replace the new key / value pair both in the volatile memory and in the INI file on the filesystem.
  - Example:

    unsigned short res = set_value(“section.color.red”, “roses are red”);
  - *Return values*: 0 in case of success, 4 in case a resource file has not been loaded yet, 255 in case of generic error.

A C++ Server application able to use the above mentioned shared library and to expose a basic API to `localhost:12345`. The server shall expose the following APIs (every API must end with a \n character):

- **`LOAD PATH`**:
  - will load the INI file specified by the PATH argument (by calling the load_resource library API).
  - *Return values*: the same returned by the library followed by a \n character, or 127\n in case of unknown command
  - Example: "***LOAD /tmp/example.ini\n" -> "0\n***"

- **`GET KEY`**:
  - will get the value identified by the KEY argument (by calling the get_value library API).
  - *Return values*: the same returned by the library followed by the loaded value and by a \n character, or 127\n in case of unknown command.
  - Example: "*`GET section.foo.bar\n" -> "0 some value\n"`*
- **`SET KEY VALUE`**:
  - will set the value identified by the VALUE argument at the KEY argument (by calling the set_value library API).
  - *Return values*: the same returned by the library followed by a \n character, or 127\n in case of unknown command.
  - Example: "*`SET section.color.red roses are red\" -> "0\n`*".

A C++ Client application exposing a user a basic CLI allowing a user to request the server to perform the above TCP requets to the Server:

- *`./client --load /tmp/example.ini`* -> triggers the LOAD PATH Server API and prints the results to standard output
- *`./client --get section.foo.bar`* -> triggers the GET KEY Server API and prints the results to standard output
- *`./client --set section.color.red "roses are red"`* -> triggers the SET KEY VALUE Server API and prints the results to standard output.

A simple bash script to be used to test the system:

- Test 1
  - Launch the server.
  - Verify the server is up and running by checking its PID and the servers listening to port 12345.
  - Stop the server with a SIGINT unix signal.
  - Verify the server is no more running by checking its PID and that there are no more servers listening to port 12345.

- Test 2
    - Write a test INI file to /tmp.
    - Launch the server
    - Launch the client and load the test INI file
    - Verify the load succeeded.
    - Launch the client and get one of the values inside the test INI file.
    - Verify the get operation succeeded.
    - Launch the client and get a non existent value.
    - Verify the operation failed with error 3.
    - Launch the client and set a new key value pair.
    - Verify the set operation succeeded.
    - Stop the server with a SIGINT unix signal.

- Effective usage of the latest C++ standards would be appreciated (at least C++11).
- The C++ applications must be built using at least a basic Makefile, but using a CMake file would be appreciated.
- The C++ applications should log their behavior at least to the standard error, but using a proper logging library would be appreciated.
- The C++ Server application should be able to serve at least one Client application at time

Simplified INI file reference:

```
[section]

foo.bar = some value
```