**Simulation of Rock Paper Scissors with a Multi threaded application**

This is a simulation of the famous Rock-Paper-Scissors tie breaker game between two players. As per the requirement, this is the design of the application.

There are 3 threads running as part of the executable:

1. One main thread which acts as a server issuing game start commands and making decisions on the winner and maintaining the history of the games.
2. Two threads – one per player to model the two players

Main thread creates server socket to handle client connections and for communication between the threads. Game decisions and JSON serialization are some of the other responsibilities handled in the main thread.

Player threads have two main functions – connect to the server socket, wait for server commands and apply the player strategy and send the response back to main thread.

JSON parser and serialization requirement has been met with the output of the complete game history serialized into **result.json** file.

Limitations of the application due to time crunch:

1. Test app is NOT written yet to validate JSON output file
2. Third party JSON library used here is [jsoncpp](https://github.com/open-source-parsers/jsoncpp) – which internally uses an std::map to represent key:value pairs. Hence, all JSON objects have keys sorted in alphabetical order. The requirement mentions a specific order (Round -> Winner -> Inputs) in which JSON object is to be constructed. The order of values in JSON should NOT matter ideally but given more time, I can try out and explore some other third party JSON libraries and get this working as per the exact requirement.
3. Though I am closing both client and server sockets, sometimes the TCP connection goes to TIME\_WAIT state. When the application is restarted, the connect will fail if TIME\_WAIT has NOT expired yet on the previous connection. Optimization can be to bind to a different client port and connect.

**Tool chain and platform:**

Cygwin - CYGWIN\_NT-10.0 2.8.2(0.313/5/3) x86\_64

GCC / G++ - 6.3.0

Jsoncpp

CMAKE for building JSONCpp

**Compilation and execution:**

Compilation – cd <project root directory> and run “**make**”

Execute - ./**game.exe > out.txt** – out.txt for console output and **result.json** which has the entire game history generated in the project root directory

**Submission:**

Complete source code with jsoncpp static library and header files have been submitted.

For build and execute, please follow the instructions above on Cygwin or a linux platorm.