Report for Client-Server-Temperature

Design:

- Winsock for client server connection using C++
    - Winsock is an API used to handle input/output with network interfaces
    - It is based on TCP/IP protocol
    - **Server Side:**
    - It creates a socket on the server side to allow client to connect
        - `socket(AF_INET, SOCK_STREAM, 0)`
    - Binds the socket created to an ip address and port so the client can listen on that port and connect to server at said port
        - `bind(listening, (sockaddr*)&hint, sizeof(hint))`
    - Tells winsock the socket is ready for listening
        - `listen(listening, SOMAXCONN);`
    - Client accepts connection to socket
        - `accept(listening, (sockaddr*)&client, &clientSize)`
    - Server sends temperature readings (random numbers) in array of characters buf to client
        - `send(clientSocket, buf, 3, 0)`
    - **Client side:**
    - Connects on the port number used in server where socket is at
        - `connect(sock, (sockaddr*)&hint, sizeof(hint))`
    - Buf fills with temperature readings send from server to client
        - `recv(sock, buf, 3, 0)`
    - Chrono library is used to track the time between each calculations to produce after 5 seconds
        - `chrono::steady_clock sc;`
    - Buf is converted into integer so average and accumulation over time is calculated
        - `auto time_span = static_cast<chrono::duration<double>>(end-start);`
    - Design choice : Winsock was used to allow communication between Windows network and other network services which allowed me to make windows programs and TCP/IP services to work together easier.
    - Reference : https://bitbucket.org/sloankelly/youtube-source-repository/src/master/cpp/networking/

- Makefile for automated build
    - Makefile includes a set of rules for automating the building procedure
    - It includes dependencies and shell commands to execute build
    - build_ all builds .cpp and .exe files using g++ that compiles C++ programs
        - `$(CC) -o ./build/client.exe ./client/client.cpp ${CFLAGS}`
    - run_server & run_client runs executable files produced
        - `./build/server.exe`
    - **Commands for build:**
        - make build_all
        - make run_server
        - make run_client
    - **Implementation choice**: Makefile provides a simple way to specify dependencies and parses these dependencies and run the associated build actions for a simpler compiling process.
    - `Reference: https://www.techbeamers.com/makefile-tutorial-create-client-server-program/`