**Project 2 report**

**Summary:**

In this project we designed a server socket and a server client we first sent a file, downloaded the file, and then we compared the original sent with the downloaded.

**Details about the program design:**

**client.py:**

 Functionality: The program is designed to perform several key functions:

* Upload a file from the client to the server.
* Download a file from the server to the client.
* Compare the downloaded file with the original file.
* Display a success or failure message based on the comparison.

 Modular **Design**: The program is organized into separate functions, which is a good practice for modularity and maintainability. Each function has a specific purpose: upload, download, compare, and result.

 Socket **Communication**: The program uses Python's socket library to establish a connection with a server running on "localhost" at port 8080. It then uses this connection to send and receive file data.

 File **Handling**: The program efficiently handles file data by reading and writing files in binary mode ("rb" and "wb") to ensure the correct transfer of binary data.

 Comparison **Logic**: The compare function compares two files by reading their binary content and checking for equality. It returns a Boolean value indicating whether the files match.

 Success **Reporting**: The result function takes the result of the comparison and prints a message to the console, indicating whether the transmission was successful or not.

 Error **Handling**: It's important to note that this program lacks error handling for potential issues during the upload, download, or file comparison processes. You might want to consider adding error-handling code to address possible exceptions that could occur.

**server.py:**

 Functionality: The program serves as a server application with the following key functions:

* Listens for incoming connections from clients.
* Receives a file from a connected client.
* Sends a file to a connected client.

 Modular **Design**: The program is organized into several functions, which enhances modularity and code readability. The functions include receive, send, and interrupt.

 Socket **Communication**: The program uses Python's socket library to establish a server socket and listen for incoming connections on "localhost" at port 8080. It also establishes a client socket when a client connects.

 File **Handling**: File data is efficiently handled using binary file read and write operations ("wb" and "rb") to ensure correct file transfer of binary data.

 Signal **Handling**: The program uses the signal module to handle the SIGINT signal (typically generated by pressing Ctrl+C). When a SIGINT signal is received, the interrupt function is called, which allows the server to close the server socket and exit gracefully.

 Error **Handling**: The program includes some error handling by closing the server socket in the interrupt function if it's open. This helps avoid resource leaks and ensures a clean shutdown.

 Console **Output**: The program provides informative console output, such as messages indicating the server's status, connection establishment with clients, and file transfer progress.

 Continuous **Listening**: The program employs a continuous listening loop (while True) to keep the server running and ready to accept connections from clients.