# Write Up: Assignment 1
# CSE130: Principles of Computer System Design

Rahul Mahendru

## 1 Testing

The program was tested in three phases.

1. Unit testing: While developing code, in order to check individual functions and cases required to achieve optimum functionality, every function and line of code was tested individually to ensure required results are achieved. This was done by printing values and using the `curl` functionality to test every function and the outputs.

2. Testing file handling for .txt files, .bin files by passing zero bytes, little amount of data, and large chunks of data for both the cases.

3. Testing for error by changing file permissions using `chmod` for the requests.

4. Using the `UNIX diff` method to check for differences in the files created by using the `GET, PUT, HEAD` functions.

5. Testing the system as a whole by sending multiple requests to locate errors and put test cases to handle errors.

## 2 Questions

**What fraction of your design and code are there to handle errors properly? How much of your time was spent ensuring that the server behaves "reasonably" in the face of errors?**
In order to handle errors, the design occupies a comparatively smaller proportion than code itself. I may say that around 50% of the code is dedicated to handling errors. The mismatch in ratio arises from the fact that while i was creating the design document, i wasn't sure about how a lot of functionality would look like in code. While testing each function, new errors arose which led to addition of error handling in both the design and for subsequent code. It only took me around an hour to implement the basic functionality of the code. However it took me another round of restructuring the code to

handle all the errors. This was due to the fact that i missed very important specifications to handle errors while only focusing on a very specific case of shorter data to implement functionality.

**List the "errors" in a request message that your server must handle. What response code are you returning for each error?**
The server is currently handling errors that may arise due to the passage of an inappropriate method, which would be anything other than `PUT, GET and HEAD`. The server also checks for a valid filename according the program specification and this is followed by checking for a valid HTTP version.

**What happens in your implementation if, during a PUT with a Content-Length, the connection is closed, ending the communication early?**
In this case, the client does not send the data from the file the client is trying to send. Therefore no bytes are sent from the client to the server in terms of data. This is an example of a case when the client requests to `PUT` a file to the server but an error occurs in terms of writing permissions or or other errors.

**Does endianness matter for the HTTP protocol? Why or why not?**
Yes, endianness does matter for a HTTP protocol. This is because the protocol uses bytes for communication between the server and the client. The bytes must be converted into network byte order before communication can take place. This is done using `htons()` to convert from host to network short and `htonl()` to convert from host to network long.