Week 4: Sockets

Exercise Instructions:

During this exercise, document your progress using screenshots of the entire screen to demonstrate that you have completed all the required tasks. Save all screenshots in a folder named “proof” within your GitHub project, and organize them chronologically (e.g., 1.png, 2.png, etc.). Screenshots can be in any standard image format (e.g., png, jpeg).

You don’t need hundreds of screenshots, but please include enough to clearly show completion of the required tasks. Your README file must also include a link to the repository.

For Submission:

Download the entire repository from GitHub and upload it to Moodle. Do not provide links to external locations (e.g., Google Drive or GitHub links). Note: Moodle has a 5MB upload limit. If necessary, remove screenshots from the submission to fit the size limit, but ensure that all screenshots remain in the GitHub repository.

1 Sockets

- Execute the code files which are attached in the Moodle to this mini-exercise, which are based on the following lecture slides: 42, 43, 44, 49, 51, 53, 55, 56. (The slide number is written **on the slide itself, not** what the PDF reader is counting as a “page”.)
- You can execute the code using two terminals, and can even run it in your favorite IDE, just make sure you know how to use two terminals in the IDE.
- When you execute the code, you execute client and server together.
- Make sure you understand the code.
- Explain the code in your own words. In the Python code, explain every line of code. In C
  Cpp, every few lines of code.

- Make a few changes to the code and explain these changes.

2 Simple web server

The following is a Python code of a *server*, which is very similar to the TCP Python server code from the lecture. It supports an application layer protocol called HTTP (which we will learn about in the future).

```
from socket import socket, AF_INET, SOCK_STREAM

s = socket(AF_INET, SOCK_STREAM)
s.bind(('', 12345))
s.listen(5)

while True:
    client, addr = s.accept()

    request = client.recv(4096)
    print(request)

    client.send(b'HTTP/1.1 200 OK\n')
    client.send(b'Content-Type: text/html\n')
    client.send(b'Access-Control-Allow-Origin: *\n')
    client.send(b'\n')
    client.send(b'Hello world ')
    client.send(b'<b>Hello world</b> ')
    client.send(b'<u>Hello world</u> ')

    client.close()
```

1. Execute the code and use a browser to access: http://localhost:12345 and ensure that you get hello world (three times).

2. What just happened?! You created a **web server**!

3. Your browser opened a TCP connection to port 12345 (which your Python code has a **listening** socket on it) and they communicated using a protocol called **HTTP**. Your browser **requested** a **Web page** and your Python **server** responded with a page that contains: "Hello world" (three times), and that is what the browser displayed.

4. Read the code.