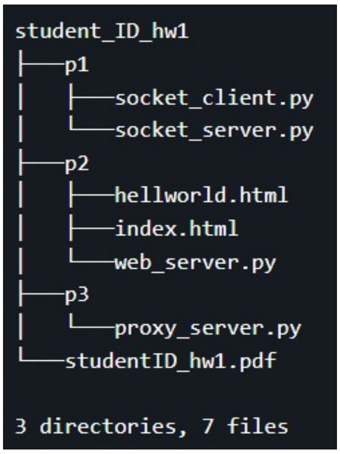
Homework1

Due date: 2022/10/31 21:00

Homework Policy: (Read before you start to work)

1. 作業請勿抄襲。疑似抄襲，助教會請你說明。如果認定抄襲，作業以零分計算。
2. 如果作業上遇到困難可以討論，但是程式碼的部分請自己完成，並且在程式第一行註明討論同學的姓名及學號。
3. 程式作業請於期限內至 NTU Cool 作業區上傳，格式為 zip 檔，解壓縮後應恰為一個以學號 (英文字母小寫) 為名的資料夾，資料夾內有三個子資料夾與相對應的檔案，以及一個以自己學號為名的 pdf 檔 ([student_id]_hw1.pdf)，如下所示：



```
student_ID_hw1
├──p1
│   ├──socket_client.py
│   └──socket_server.py
├──p2
│   ├──hellworld.html
│   ├──index.html
│   └──web_server.py
├──p3
│   └──proxy_server.py
└──studentID_hw1.pdf

3 directories, 7 files
```

Figure 1: Folder Structure.

4. 逾期繳交一天，分數 ×$\frac{2}{3}$；超過一天未滿兩天，分數 ×$\frac{1}{3}$；超過兩天則不予計分，請務必盡早開始，並努力完成。
5. 若助教無法編譯你的程式，以 0 分計算。繳交檔案格式錯誤會扣 10 分。
6. 在測試自己的程式碼時，請使用(自己學號末四碼 + 1023) % 65535 作為 port number。例如：學號為 b10901777，port number 即為 (1777+1023)%65535=2800。
7. 如有任何問題歡迎來信，請同時寄信給兩位助教。
   範例: [2022ICN] HW1
   信箱: 林致佑 d09921030@ntu.edu.tw、陳嘉宏 r11921051@ntu.edu.tw

Problems:

1. **Socket Programming - TCP** – 30%
   We are going to construct a TCP server and client system. Figure 2 describes the messages sent between client and server.

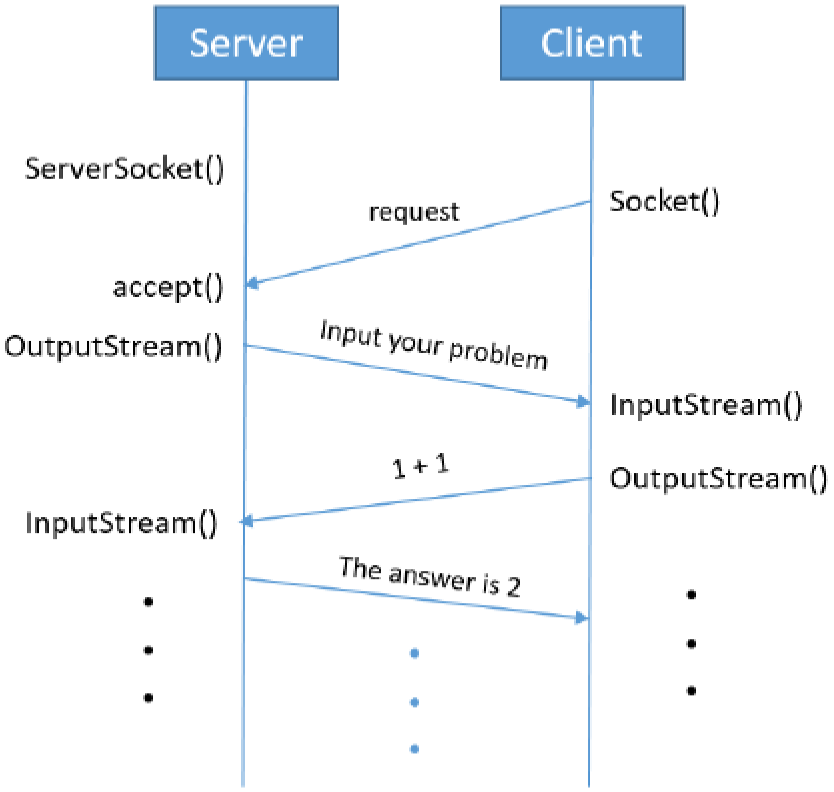


Figure 2: Message flow between client and server

Please refer to the following steps:

(a) Read the source of socket_server.py (works on python3 environment), try your best to understand it.

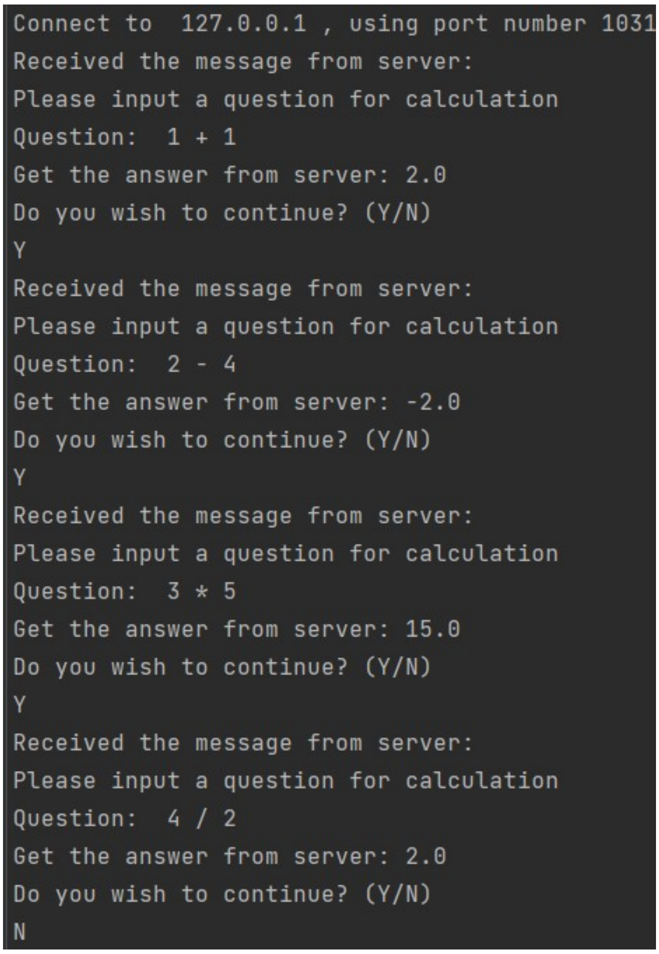
(b) Complete ”TODO” part at socket_server.py. Let the server to work as a calculator that supports basic ”+”,”-”,”×”,”÷” operations.

(c) Complete ”TODO” part at socket_client.py such that it can (1) read the message sent from server and (2) send the user-input message to the server to check whether the server works as you expect. The message flow between client and server is shown in Figure 2.

(d) Compile and run the socket_server.py first, and then run your socket_client.py.

(e) Test it under your local machine. If you do it correct, the execution and output of socket_client.py should be like Figure 3.

(f) When you finish (e), try to run your socket_client.py and connect to TA’s computer with TA’s IP address 140.112.42.104 and port 7777.



```
Connect to  127.0.0.1 , using port number 1031
Received the message from server:
Please input a question for calculation
Question:  1 + 1
Get the answer from server: 2.0
Do you wish to continue? (Y/N)
Y
Received the message from server:
Please input a question for calculation
Question:  2 - 4
Get the answer from server: -2.0
Do you wish to continue? (Y/N)
Y
Received the message from server:
Please input a question for calculation
Question:  3 * 5
Get the answer from server: 15.0
Do you wish to continue? (Y/N)
Y
Received the message from server:
Please input a question for calculation
Question:  4 / 2
Get the answer from server: 2.0
Do you wish to continue? (Y/N)
N
```

Figure 3: Connect to server on local machine (127.0.0.1).

(g) (Bonus 5%) You can also implement other math functions. For example, matrix multiplication, integral, or Laplace's transform. Each function you add, you get 1 extra point in this problem. However, the existing math libraries are not allowed. You should write these steps by your own and explain how TA can test your functions in [student_id]_hw1.pdf.

Submission and Grading Policy

(a) Please put your source code (including socket_server.py and socket_client.py) under [student_id]/p1. If you have done extra functions in (g), you should also give brief explanation in [student_id]_hw1.pdf.

(b) We provide p1_testcase for you to test your program. The result should be the same as p1_testcase_golden. Please make your program available to save the result to a file [student_id]_p1_client_result.log. TA will use this file to grade your program.

(c) Each operator will take 5 points. You can get the other 10 points by implementing the socket. We will test your client with your server and our server both.

2. **Web Server** – 25%

   (Ch.2 Socket Programming Assignment#1,#4 in textbook)
   In this assignment, you will develop a simple Web server in python that is capable of processing simple request. There should be two HTML files. One is [index.html] and the other is [helloworld.html]. You can access the two HTML files through the Web server. Also, you can access [helloworld.html] through [index.html].

   Please refer to the following steps:

   (a) Run the Web server on the local machine and create a connection socket.

   (b) Create two HTML files as described above.

   (c) Receive the HTTP request from this connection when contacted by a client (browser).

   (d) Parse the HTTP request to determine the specific file being requested.

   (e) Get the requested file from the server's file system.

   (f) Create an HTTP response message consisting of the requested file preceded by header lines.

   (g) If the client sends HTTP request for accessing other pages through homepage (index.html), the Web server will deal with the request as the same from step (c) to step (f).

   (h) Send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in your server, your Web server should return a "404 Not Found" error message. The message flow between client and web browser and Web server is shown in Figure 4.

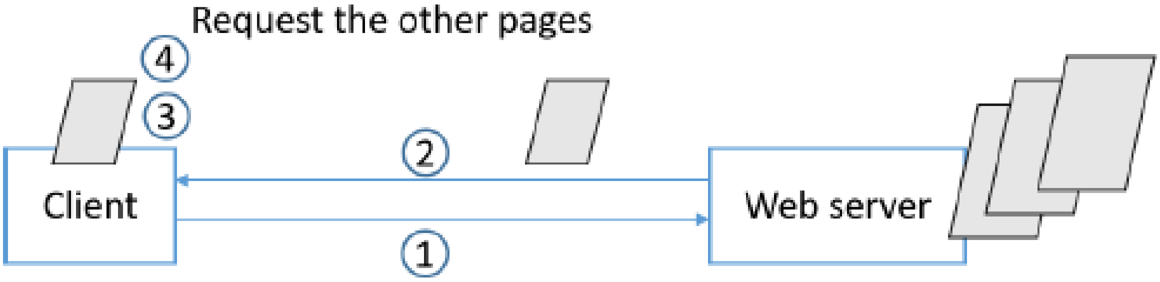


Figure 4: Message flow between the client and the Web server.

Notes

(a) Read the source code web_server.py, and try your best to understand it. Complete "TODO" part for Web server functions.

(b) Be sure to put HTML files under the directory that Web server is running, so you can can get HTML files through the browser.

(c) Get IP address of your Web server, and try to connect to the Web server by other devices instead of the local machine. Check if you can get the file through browser in another device.

(d) Show ”Hello World!” in [helloworld.html]. Show your name and student ID in [index.html].

(e) You can ignore the error message from ”FAVICON.ICO”.

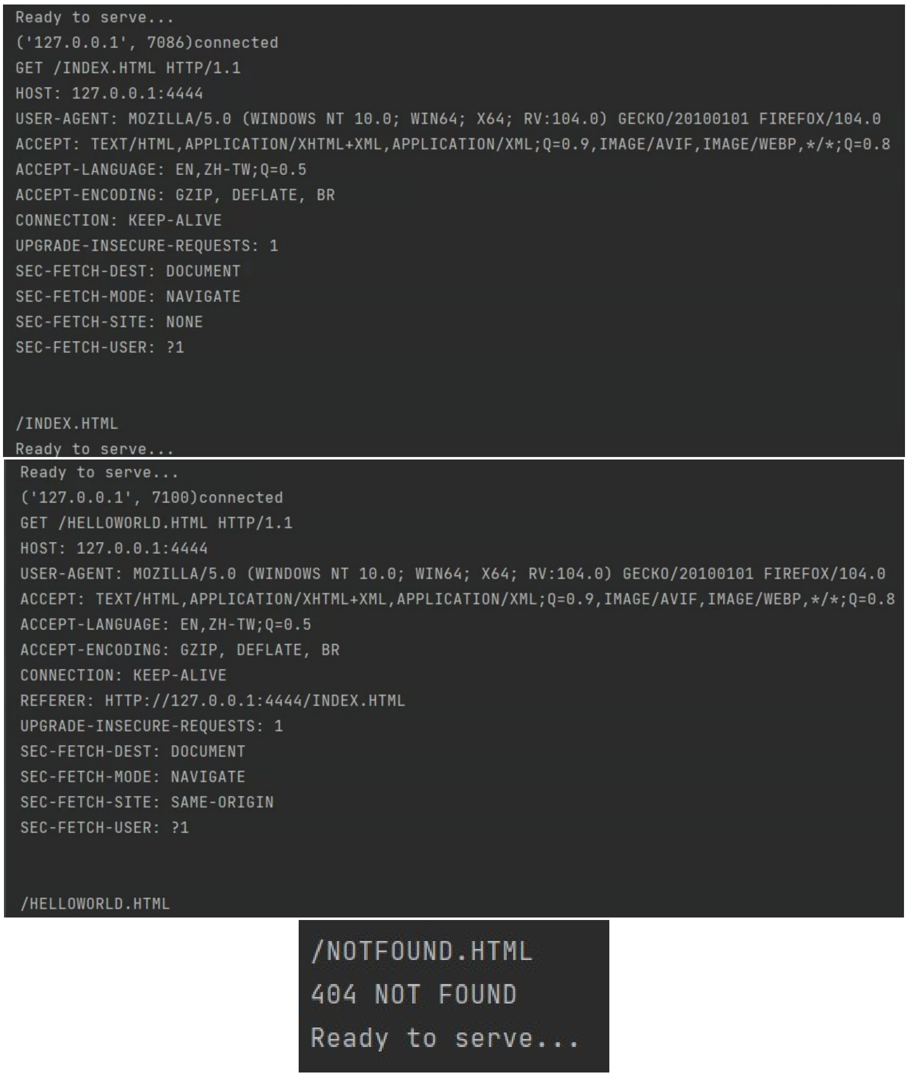
Submission and Grading Policy

(a) Please put your source code (web_server.py) and HTML files under [student_id]/p2. Also, please screenshot your output results and put them into [student_id]_hw1.pdf.

(b) Get 15 points if your program can access html files through a local machine and other devices.

(c) Get 5 points if your program can show ”404 Not Found” when the requested file is not in the Web server.

(d) Get 5 points if the content of two HTML files is correct and you can access [helloworld.html] through [index.html].



```
Ready to serve...
('127.0.0.1', 7086)connected
GET /INDEX.HTML HTTP/1.1
HOST: 127.0.0.1:4444
USER-AGENT: MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:104.0) GECKO/20100101 FIREFOX/104.0
ACCEPT: TEXT/HTML,APPLICATION/XHTML+XML,APPLICATION/XML;Q=0.9,IMAGE/AVIF,IMAGE/WEBP,*/*;Q=0.8
ACCEPT-LANGUAGE: EN,ZH-TW;Q=0.5
ACCEPT-ENCODING: GZIP, DEFLATE, BR
CONNECTION: KEEP-ALIVE
UPGRADE-INSECURE-REQUESTS: 1
SEC-FETCH-DEST: DOCUMENT
SEC-FETCH-MODE: NAVIGATE
SEC-FETCH-SITE: NONE
SEC-FETCH-USER: ?1


/INDEX.HTML
Ready to serve...
```

```
Ready to serve...
('127.0.0.1', 7100)connected
GET /HELLOWORLD.HTML HTTP/1.1
HOST: 127.0.0.1:4444
USER-AGENT: MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:104.0) GECKO/20100101 FIREFOX/104.0
ACCEPT: TEXT/HTML,APPLICATION/XHTML+XML,APPLICATION/XML;Q=0.9,IMAGE/AVIF,IMAGE/WEBP,*/*;Q=0.8
ACCEPT-LANGUAGE: EN,ZH-TW;Q=0.5
ACCEPT-ENCODING: GZIP, DEFLATE, BR
CONNECTION: KEEP-ALIVE
REFERER: HTTP://127.0.0.1:4444/INDEX.HTML
UPGRADE-INSECURE-REQUESTS: 1
SEC-FETCH-DEST: DOCUMENT
SEC-FETCH-MODE: NAVIGATE
SEC-FETCH-SITE: SAME-ORIGIN
SEC-FETCH-USER: ?1


/HELLOWORLD.HTML
```

```
/NOTFOUND.HTML
404 NOT FOUND
Ready to serve...
```

Figure 5: Sample output of problem p2.

3. **Proxy Server** – 35%

   In this assignment, you will develop a Web proxy. The message flow is shown in Figure 6. Followings are the things that you have to do.

   (a) Create a socket on the proxy server and receive data (request) from the client.

   (b) Parse the request to determine the which file is being requested.

   (c) Check whether the requested file in the local server. If yes, send the file back as you did in p2. If not, go to step(d).

   (d) Create another socket on the proxy server and connect it to the Web server (the same step as you did in p2).

   (e) Ask the Web server for the file requested by the client.

   (f) Receive and read the response, and then send it back to the client. Also, cache the file in local proxy server.

   (g) If the file is not found, return an error message ”404 NOT FOUND”.

   Notes

   (a) Read the source code proxy_server.py, and try your best to understand it. Complete ”TODO” part for proxy server functions.

   (b) Test your program by using different browsers to request web objects through your proxy.

   (c) Be sure to configure your web browser to use the proxy at first.

   (d) You can ignore the error message from ”FAVICON.ICO”.

   Submission and Grading Policy

   (a) Please put your source code (proxy_server.py) under [student_id]/p3. Also, please screenshot your output results and put them into [student_id]_hw1.pdf.

   (b) Get 25 points if your program can access HTML files (from p2) through the proxy server.

   (c) Get 5 points if your program can show ”404 Not Found” when the requested file is not in neither the proxy server or the Web server.

   (d) Get 5 points if you can access HTML files (from p2) through two different browsers.

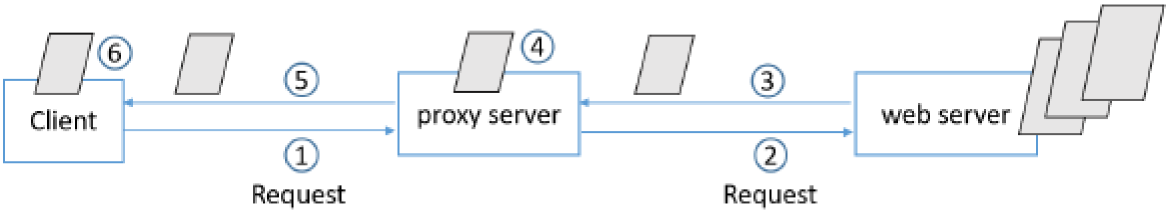
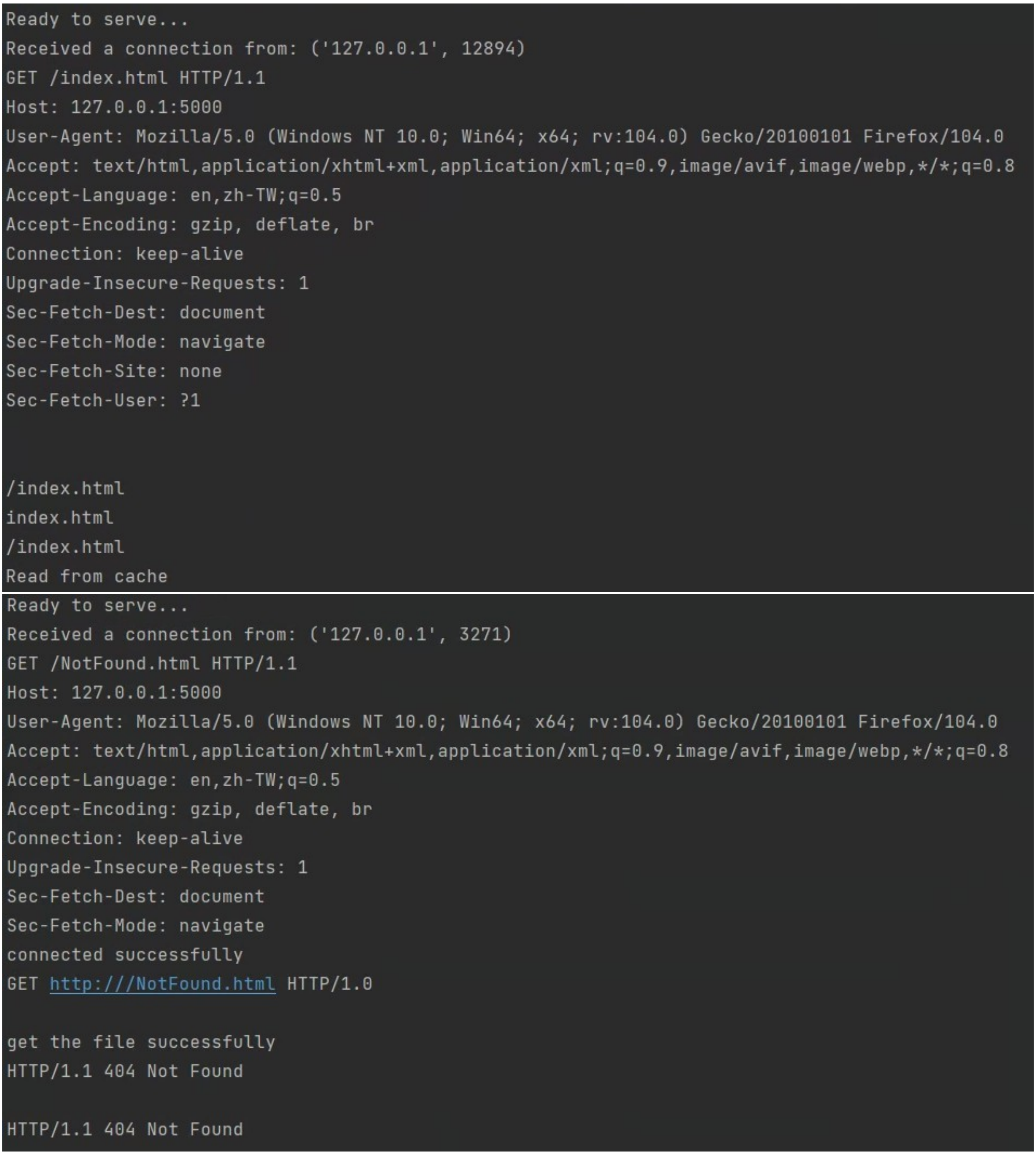


Figure 6: Message flow among Client-Proxy-Server.



```
Ready to serve...
Received a connection from: ('127.0.0.1', 12894)
GET /index.html HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en,zh-TW;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


/index.html
index.html
/index.html
Read from cache
```

```
Ready to serve...
Received a connection from: ('127.0.0.1', 3271)
GET /NotFound.html HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en,zh-TW;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
connected successfully
GET http:///NotFound.html HTTP/1.0

get the file successfully
HTTP/1.1 404 Not Found

HTTP/1.1 404 Not Found
```

Figure 7: Sample output of problem p3.

Report ([student_id]_hw1.pdf) – 10%

1. If you have done the bonus part of p1, briefly explain your additional functions and how to test your implementations.
2. (a) Post your output results.
   (b) Get 5 points if you briefly describe how you implement ”TODO” part function in p2.
3. (a) Post your output results.
   (b) Get 5 points if you briefly describe how you implement ”TODO” part function in p3.