Policies and Guidelines

- Assignments deadline come with a ***72 hours grace period***. If you submit the assignment during the grace period, it will be accepted with a ***20% penalty***. Submissions after the grace period will be accepted with a ***50% penalty***. ***No submission will be accepted after the last day of classes on Thursday, May 4, 2023, at 11:59 pm***.
- Every project has required submission guidelines. Please read the submission requirements carefully. Any deviations from specifications on projects will result in point deductions or incomplete grades.
- ***Instructors will not 'fix' your code to see what works***. If your code does not run due to any type of errors, it is considered non-functioning.
- ***You cannot resubmit 'fixed' code after the deadline***, regardless of how small the error is.
- If you use the University resources for development and/or testing, then keep in mind that others in the University may need to use the same resources, so please use these resources wisely. You should also follow the rules and restrictions associated with using the University resources.
- Using any external/third-party library and/or framework to implement the project requirements is not permitted.

Academic Honesty Expectations and Violation Penalty

- ***The programming assignments are team assignments; each team cannot have more than two members***. Each team must do the vast majority of the work on its own. It is permissible to consult with other teams to ask general questions, help discover and fix specific bugs, and talk about high-level approaches in general terms. ***Giving or receiving answers or solution details from other teams is not permissible***.
- You may research online for additional resources; however, ***you may not use code explicitly written to solve the problem you have been given***. ***You may not have anyone else help you write the code or solve the problem***. You may use code snippets found online, providing that they are appropriately and clearly cited within your submitted code.
- ***If you use a distributed version control service such as GitHub to maintain your project, you should always set your project repository to private***. If you make your project repository public during the semester or even after the semester, it may cause a violation of the academic honesty policy if other students find and use your solution.

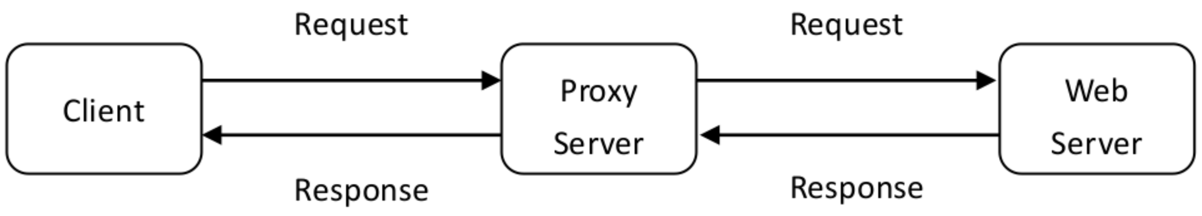
- If you are involved in plagiarism or cheating, you will receive a score of "0" for the involved project for the first offense. You will receive an "F" grade for the course and be reported to the university for any additional offense.
- Students are required to strictly follow the rules and guidelines laid out in the Watson School Student Academic Honesty Code at the following link: http://www.binghamton.edu/watson/about/honesty-policy.pdf
- Please also review Computer Science Faculty Letter to Students Regarding Academic Honesty at the following link: http://www.cs.binghamton.edu/~ghyan/courses/CSHonestyLetterToStudents.pdf
- Cheating and copying will **NOT** be tolerated. Anything submitted as a programming assignment must be the student's original work.
- We reserve the right to use plagiarism detection tools to detect plagiarism in the programming assignments.

Implementation

Part 1: Web Proxy Server

In this lab, you will learn how web proxy servers work and one of their basic functionalities, caching. Generally, when the client makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client. In order to improve the performance, we create a proxy server between the client and the web server. Now, both the request message sent by the client and the response message delivered by the web server passes through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client.

Extend your Project-1 multi-threaded web server implementation to support a multi-threaded web proxy server. Assume that caching is not enabled on the proxy server. In this case, the main functionality of the proxy server is to forward requests received from the client to the web server and forward responses received from the web server to the client, as shown in the figure below.



When the proxy server forwards a request to the web server or forwards a response to the client, it should print the following information on the terminal:

```
proxy-forward,DESTINATION,THREAD-ID,TIMESTAMP
```

Where "DESTINATION" is either client or server.

Also, modify your web server implementation to print the following information on the terminal whenever it responds to a request:

```
server-response,STATUS-CODE,THREAD-ID,TIMESTAMP
```

Where the "STATUS-CODE" is the HTTP response code indicating the result of the request.

Part 2: Caching

A typical proxy server will cache the web pages each time the client makes a particular request for the first time. The basic functionality of caching works as follows. When the proxy gets a request, it checks if the requested object is cached and if yes, it returns the object from the cache without contacting the server. If the object is not cached, the proxy retrieves it from the server, returns it to the client, and caches a copy for future requests. The proxy also verifies that the cached responses are still valid.

Add the simple caching functionality described above to your implementation. Assume cached responses older than 120 seconds are no longer valid and should be discarded. Your proxy implementation should record the time when a new server response is copied (cached) on the proxy. It should also be able to write responses to the disk (i.e., the cache) and fetch them from the disk when it gets a cache hit. For this, you need to implement some internal data structure in the proxy to track which objects are cached and where they are on the disk. You can keep this data structure in the main memory; it does not need to persist across shutdowns.

When there is a cache hit in the proxy, and it is valid, the proxy should print the following information on the terminal after sending the cached response to the client:

```
proxy-cache,client,THREAD-ID,TIMESTAMP
```

You should implement a way to terminate the servers' main thread safely. To achieve this, you should ensure all TCP connections are closed, and all threads are completed. Also, your web and proxy servers should support transmitting and caching HTML and PDF files. When testing locally on the same device, the web server and proxy server sources should be placed in and run from separate directories.

Submission

- Complete the "README.md". You may write “n/a” where applicable (bugs, references, etc.).
- Please submit the following files for Project-2 on Brightspace using the same file name and extension as here:
  - webserver.py (part 1)
  - proxyserver1.py (part 1)
  - proxyserver2.py (part 2)
  - Screenshots (PNG) at the client, proxy server, and web server for part 1 and part 2 verifying that the terminals print the correct output and the browsers render the requested files correctly.
  - README.md
- You must submit your project before the deadline to be considered on time. Otherwise, it will be considered late even if your project was completely working before the deadline. ***Please note that on Brightspace, we maintain all your submissions but only grade your most recent submission.***

Grading Rubric

Total: 100 points

- Part 1: 35 points
  - Demonstration: 15 points
  - Implementation: 20 points
- Part 2: 50 points
  - Demonstration: 25 points
  - Implementation: 25 points
- Submission: 15 points
  - README.md: 10 points
  - All other requirements: 5 points
- If submission didn’t pass the plagiarism test(s): -100 points.