Performance Experiments:-

1. Functionality check:- If my server and client are in different machines, it is giving correct responses.

2. Response Time check:- Do multiple runs and note the time for the response.

3. Throughput check: How many grading requests per second does the server support?

==> **Average Response Time: .4801 seconds**
**Throughput: 2.0828 requests/second**

**4.** Concurrency check: What happens if you start some N (large) clients, all sending files to the server to grade?

```
1 2 3 4 5 6 7 8 9 10

PASS
1 2 3 4 5 6 7 8 9 10

PASS
1 2 3 4 5 6 7 8 9 10

Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
Connection error: Connection timed out
PASS
1 2 3 4 5 6 7 8 9 10

PASS
1 2 3 4 5 6 7 8 9 10

PASS
1 2 3 4 5 6 7 8 9 10

PASS
1 2 3 4 5 6 7 8 9 10

Error receiving response from server
Error receiving response from server
Error receiving response from server
Error receiving response from server
SERVER_IP="10.96.26.15"
PORT="8888"
SOURCE_FILE="program.cpp"

# Number of requests to send
NUM_REQUESTS=250
```

## Observations and challenges arises:-

1. **Blocking Behavior:** In a single-threaded server, it can only handle one request at a time. As a result, clients beyond the first will have to wait in a queue until the server finishes processing the previous request. This can lead to significant delays for clients, especially if N is large.
2. **Increased Response Time:** As more clients send grading requests, the response time for each individual client will increase. Each client must wait for its turn to be processed by the server, leading to longer response times.
3. **Resource Contention:** The single-threaded server may experience resource contention, such as CPU and memory usage, as it attempts to handle all the incoming requests. This contention can slow down the server's processing speed.
4. **Concurrent Requests Queuing:** Clients may experience queuing delays, where their requests are placed in a queue as they wait for the server to become available. The length of the queue can grow significantly with a large number of clients.
5. **Potential for Denial of Service (DoS):** If the number of clients attempting to connect to the server exceeds its capacity to handle them, it can lead to a situation where legitimate clients are unable to connect or experience excessive delays. This situation can be exploited for a denial of service attack.
6. **Server Saturation:** Beyond a certain point, if the number of clients is excessively large, the server may become saturated and struggle to keep up with the incoming requests. This can lead to server crashes or unresponsiveness.
7. **Concurrent File Handling:** If clients are sending files to be graded, the server must also manage concurrent file access and potential file conflicts. Without proper synchronization, file handling issues may arise.
8. **Scalability Limitations:** A single-threaded server has inherent limitations in terms of scalability. It cannot efficiently utilize multi-core processors, which limits its ability to handle large numbers of concurrent clients.