CN Assignment #1

Code:

import socket

import sys

import multiprocessing

MAX\_CONNECTIONS = 100

BUFFER\_SIZE = 4096

def receive\_request(client\_socket):

    try:

        request\_data = client\_socket.recv(BUFFER\_SIZE).decode()

        return request\_data if request\_data else None

    except Exception as err:

        print(f"[Error receiving request] {err}")

        return None

def parse\_request(request\_data):

    try:

        request\_lines = request\_data.split("\r\n")

        request\_line = request\_lines[0].split()

        if len(request\_line) != 3:

            return None, None, None, None, "HTTP/1.0 400 Bad Request\r\n\r\n"

        method, full\_url, http\_version = request\_line

        if method != "GET":

            return None, None, None, None, "HTTP/1.0 501 Not Implemented\r\n\r\n"

        if full\_url.startswith("http://"):

            full\_url = full\_url[7:]

        host\_part, \_, resource\_path = full\_url.partition("/")

        resource\_path = "/" + resource\_path if resource\_path else "/"

        if ":" in host\_part:

            host, port = host\_part.split(":")

            port = int(port)

        else:

            host, port = host\_part, 80

        return method, host, port, resource\_path, None

    except Exception as err:

        print(f"[Error parsing request] {err}")

        return None, None, None, None, "HTTP/1.0 400 Bad Request\r\n\r\n"

def forward\_request(host, port, resource\_path, request\_lines, client\_socket):

    try:

        with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

            server\_socket.connect((host, port))

            request\_to\_server = f"GET {resource\_path} HTTP/1.0\r\n"

            request\_to\_server += "\r\n".join(request\_lines[1:]) + "\r\n\r\n"

            server\_socket.send(request\_to\_server.encode())

            while True:

                response = server\_socket.recv(BUFFER\_SIZE)

                if not response:

                    break

                client\_socket.send(response)

    except Exception as err:

        print(f"[Error forwarding request] {err}")

    finally:

        client\_socket.close()

def process\_client(client\_socket):

    request\_data = receive\_request(client\_socket)

    if not request\_data:

        client\_socket.close()

        return

    method, host, port, resource\_path, error\_response = parse\_request(request\_data)

    if error\_response:

        client\_socket.send(error\_response.encode())

        client\_socket.close()

        return

    forward\_request(host, port, resource\_path, request\_data.split("\r\n"), client\_socket)

def start\_proxy(port):

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as proxy\_socket:

        proxy\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

        proxy\_socket.bind(("0.0.0.0", port))

        proxy\_socket.listen(MAX\_CONNECTIONS)

        print(f"Proxy server running on port {port}...")

        while True:

            client\_socket, client\_addr = proxy\_socket.accept()

            print(f"New connection from {client\_addr}")

            process = multiprocessing.Process(target=process\_client, args=(client\_socket,))

            process.start()

            client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

    print("Name: Ausaja")

    print("Roll Number: 22K-5186")

    print("Section: BSE-6B\n\n")

    if len(sys.argv) != 2:

        print(f"Usage: {sys.argv[0]} <port>")

        sys.exit(1)

    proxy\_port = int(sys.argv[1])

    start\_proxy(proxy\_port)

Output:



