/e2/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    # Assume the path is a directory or file name.
    # If the directory or file exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
...
```

/e2/my_http_server.py

Question 2

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
       # Handle directory requests.
        if os.path.isdir(path):

            # If directory does not end with '/', send a 301 with the correctly formed URL.
            # Otherwise, send the directory normally.
            if not path.endswith("/"):
                socket.send_text_line("HTTP/1.0 301 Moved Permanently")
                socket.send_text_line(f"Location: /{path}/")
            else:

                # If the directory contains index.html, return it if it is readable.
                # Otherwise, return a directory listing with links.
                if os.path.isfile(path + "index.html") and os.access(path + "index.html", os.R_
                    file_size = os.path.getsize(path + "index.html")
                    with open(path + "index.html", 'rb') as file:
                        socket.send_text_line("HTTP/1.0 200 OK")
                        socket.send_text_line(f"Content-Type: text/html")
                        socket.send_text_line(f"Content-Length: {file_size}")
                        socket.send_text_line(f"Connection: close")
                        socket.send_text_line("")  # <======= This line
                        socket.send_binary_data_from_file(file, file_size)
                else:
                    message = f"<html><body><h1>{path}</h1><ul>"

                    # List contents in directory
                    contents = os.listdir(path)

...
```

/e3/my_http_server.py

Question 1

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```python
def send_directory(socket, path):
    """
    Send a directory listing to the client.
    """
    # Send a directory listing
    if path[-1] != '/':
        path += '/'
        socket.send_text_line("HTTP/1.0 301 MOVED PERMANENTLY")
        socket.send_text_line(f"Location: {path}")
        socket.send_text_line("Connection: close")
        socket.send_text_line("")
        return  # Stop further execution

    socket.send_text_line("HTTP/1.0 200 OK")

    print(f"Sending directory listing for {path}")
    print("Directory contents:")
    message = f"<html><body><h1>Directory listing for {path}</h1><ul>"
    for item in os.listdir(path):
        if item == "index.html":
            send_file(socket, path + item)
            return
        print(item)
        if os.path.isdir(item):
            message += f"<li><a href=\"{item}/\">{item}/</a></li>"
        else:
            message += f"<li><a href=\"{item}\">{item}</a></li>"
    message += "</ul></body></html>"

    socket.send_text_line("Content-Type: text/html")
    socket.send_text_line(f"Content-Length: {len(message) + 2}")
    socket.send_text_line(f"Connection: close")
    socket.send_text_line("")  # <======
    socket.send_text_line(message)
...
```

/e3/my_http_server.py

Question 2

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```python
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    if request == None:
        print("Incomplete Data, Client disconnected")
        return

    # print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        # print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path
    # print("Path:", path)

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):

...
```

/e4/my_http_server.py

Question 1

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```python
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")

    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path
    try:
        extension = path[path.index("."):]
    except:
        extension = ""

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
        # We need to know the file size so we can send
        # the Content-Length header.
...
```

/e4/my_http_server.py

Question 2

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
       if (extension == ".html" or extension == ".htm" or extension == ".txt"): # If HTML or t
            with open(path, 'r') as file:
                socket.send_text_line("HTTP/1.0 200 OK")
                socket.send_text_line("Content-Type: " + EXTENSION_MAP.get(extension))
                socket.send_text_line(f"Content-Length: {file_size}")
                socket.send_text_line(f"Connection: close")
                socket.send_text_line("")  # <=========
                html = []
                while line := file.readline():
                    html.append(line)
                html = "\n".join(html)
                socket.send_text_line(html)
        elif (extension != ""): # If image/binary file
            with open(path, 'rb') as file:
                socket.send_text_line("HTTP/1.0 200 OK")
                socket.send_text_line("Content-Type: " + EXTENSION_MAP.get(extension))
                socket.send_text_line(f"Content-Length: {file_size}")
                socket.send_text_line(f"Connection: close")
                socket.send_text_line("")
                socket.send_binary_data_from_file(file, file_size)
...
```

/e5/my_http_server.py

Question 1

-

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")

    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
            break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path) and os.path.isfile(path):
        # zk A file path can have more than one . in it.
        # This code will also break if a file has no . in it.
        file_parts = path.split(".")
        file_extension = EXTENSION_MAP["." + file_parts[1]]
        # We need to know the file size so we can send
        # the Content-Length header.
...
```

/e5/my_http_server.py

Question 2

-

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
...
```

/e5/my_http_server.py

Question 3

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
     with open(path, read_type) as file:
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line(f"Content-Type: {file_extension}")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("")
            if read_type == 'r':
                while line := file.readline():
                    socket.send_text_line(line)
            else:
                socket.send_binary_data_from_file(file, file_size)
...
```

/e6/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")

    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
            break
        print(data)

    # Extract the path from the request.
    parts = request.split()

    path = parts[1][1:]  # remove the first character of the path

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    # Source: https://www.geeksforgeeks.org/python-os-path-isfile-method/
    if os.path.exists(path) and os.path.isfile(path):

        # We need to know the file size so we can send
        # the Content-Length header.
        file_size = os.path.getsize(path)
...
```

/e6/my_http_server.py

Question 2

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
        # We search through our extension map for our the extension request of file.
        # If that file extension doesn't exist then we fall back on "text/plain"
        # Source: https://www.w3schools.com/python/ref_dictionary_get.asp
        fileType = EXTENSION_MAP.get(extensionOfFile, "text/plain")

        # Debugging: http://localhost:8534/studentData/Pictures/catTyping.gif gives "image/gif
        # print("The file type is: " + fileType + "\n")

        with open(path, "rb") as file:  # We need to add b for binary.
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line(f"Content-Type: {fileType}")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("")  # <==========

            # Read and send one line at a time.
            # (This works because this server only handles text.)
            # while line := file.readline():
            # socket.send_text_line(line)

            # This allows us to send our other file types that aren't text...
            # We put in our parameters the file and the size to send.
            # We don't need to put an if statement as look through our extension map
            # and if it's not there then it's "text/plain"
            socket.send_binary_data_from_file(file, file_size)
...
```

e7/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```python
def handle_connection(client_socket):
    """
    Handle the "conversation" with a single client connection
    """
    http_sock = http_socket.HTTPSocket(client_socket)
    request = http_sock.receive_text_line()

    if not request:
        return

    method, path, _ = request.split(" ")

    if path == "/":
        path = "/index.html"

    file_path = os.path.join(ROOT_DIRECTORY, path.lstrip("/"))

    # handles 301 redirect
    if os.path.isdir(file_path):
        if not path.endswith("/"):
            http_sock.send_text_line("HTTP/1.1 301 Moved Permanently")
            http_sock.send_text_line(f"Location: {path}/")
            http_sock.send_text_line("Connection: close\r\n")
            return
...
```

e7/my_http_server.py

Question 2

`send_text_line` adds a CR and and LF. Why did you add those after `Connection: close` but not elsewhere?

Code:

```
    if not os.path.isfile(file_path):
        content = "<html><body><h1>404 Not Found</h1></body></html>"
        http_sock.send_text_line("HTTP/1.1 404 Not Found")
        http_sock.send_text_line("Content-Type: text/html")
        http_sock.send_text_line(f"Content-Length: {len(content)}")
        http_sock.send_text_line("Connection: close\r\n")
        http_sock.send_text_line(content)
...
```

/e8/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
   socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")

    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
            break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path
    simple_directory_listing = ""  # assignment before reference to fix error

    # If the requested document is a directory and does not end with a /,
    # return a 301 and redirect to a correctly formed URL.
    # (For example, if the path is /Pictures, redirect to /Pictures/.)
    if os.path.isdir(path):
...
```

/e8/my_http_server.py

Question 2

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
        # (In other words, your href needs Images/ not just Images.)
        else:
            simple_directory_listing = "<html><body><h1>Directory Listing</h1><ul>" # create he
            for link in os.listdir(path):
                linked_path = os.path.join(path, link)
                if os.path.isdir(linked_path):
                    link += "/" # add trailing / if it is a directory
                simple_directory_listing += f'<li><a href="{link}">{link}</a></li>' # create an
            simple_directory_listing += "</ul><body></html>" # close everything up

            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line("Content-Type: text/html")
            socket.send_text_line(f"Content-Length: {len(simple_directory_listing)+2}")
            socket.send_text_line("Connection: Close")
            socket.send_text_line("")  # <==============
            socket.send_text_line(simple_directory_listing)
            return
...
```

/e9/my_http_server.py

Question 1

What would happen if you remove the line `socket.send_text_line("")`?

Code:

```
def send_content_headers(socket, status, content_length, content_type=None, location=None):
    socket.send_text_line(f"HTTP/1.0 {status.value}")
    if location and status == HTTPStatus.MOVED_PERMANENTLY:
        socket.send_text_line(f"Location: {location}")
    elif content_type:
        socket.send_text_line(f"Content-Type: {content_type}")
    socket.send_text_line(f"Content-Length: {content_length}")
    socket.send_text_line(f"Connection: close")
    socket.send_text_line("")  #<======
...
```

/e9/my_http_server.py

Question 2

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")

    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    # For subsequent requests if there was an index.html file, need base directory to resolve
    # Otherwise, the path will be invalid since it is relevant to the
    if CURRENT_BASE_DIR and not os.path.exists(path):
        path = os.path.join(CURRENT_BASE_DIR, path)

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
...
```

/e10/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
   HTTP_socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g., "GET /index.html HTTP/1.0")
    request = HTTP_socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = HTTP_socket.receive_text_line().strip()
        if (not data):
                break
        print(data)
    print('=======')



   # Split the request into parts
    parts = request.split()
    if len(parts) < 2:
        HTTP_socket.send_text_line("HTTP/1.0 400 BAD REQUEST\r\nConnection: close\r\n\r\n")
        HTTP_socket.close()
        return
    # Extract the requested path from the second part of the request
    path = parts[1]

    # Check if the path is the root "/"
    if path == "/":
        file_path = "."  # Set file_path to the current directory if the request is for root
    else:
        # Otherwise, remove the leading slash
        file_path = path.lstrip('/')


    # Check if path is directory
    if os.path.isdir(file_path):
        if not path.endswith('/'):
            response = "HTTP/1.1 301 Moved\r\n"
            response += f"Location: /{file_path}/\r\n"  # Redirect with trailing slash
            response += f"Content-Length: 0\r\n\r\n"
            connection.sendall(response.encode())
            return
...
```

/e10/my_http_server.py

Question 2

Why does your `Content-Length:` string have two sets of `\r` instead of just one?

Code:

```
        else:
            items = os.listdir(file_path)
            html_content = "<html><body><h1>Directory listing</h1><ul>"
            # Loop through the directory and get the full file paths
            for item in items:
                # If the path is a directory
                if os.path.isdir(os.path.join(file_path, item)):
                    item += "/"  # Ensure subdirectories end with "/"
                html_content += f'<li><a href= "{item}/">{item} </a></li>'
            html_content += "</ul></body></html>"
            response = "HTTP/1.1 200 OK\r\n"
            response += "Content-Type: text/html\r\n"
            response += f"Content-Length: {len(html_content)}\r\n\r\n"
            connection.sendall(response.encode() + html_content.encode())
            return
...
```

/e11/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
 socket = http_socket.HTTPSocket(connection)

   # Read and print the request (e.g. "GET / HTTP/1.0")
   request = socket.receive_text_line()
   print(f"Request: {request}")


   # Read and print the request headers
   while True:
       data = socket.receive_text_line().strip()
       if (not data) or (len(data) == 0):
               break
       print(data)
   print('=======')

   # Extract the path from the request.
   parts = request.split()
   path = parts[1][1:]  # remove the first character of the path

   # Assume the path is a file name.
   # If the file name exists, we will send it as a response.
   # Otherwise, send a 404.
   if os.path.exists(path):
       status_code = "200"
       print(f"PATH HERE: {path}")

       if os.path.isdir(path):
           if path[-1] != "/":
               path += "/"
               status_code = "301"

           if os.path.exists(f"{path}/index.html"):
               path += "index.html"
           else:
               return_directory_items(socket, path, status_code)

               socket.close()
               return

       # We need to know the file size so we can send
       # the Content-Length header.
       file_size = os.path.getsize(path)
       file_extension = os.path.splitext(path)[1]
       if file_extension not in EXTENSION_MAP:
           content_type = "text/plain"
       else:
           content_type = EXTENSION_MAP[file_extension]
...
```

/e12/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```python
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    # Extract the second part (should be the path. EX: /path)
    path = parts[1]
    # Remove the leading slash from the path and combine it with the root directory
    full_path = os.path.join(ROOT_DIRECTORY, path.lstrip('/'))

    # Check if the path is a dir
    if os.path.isdir(full_path):
        if not path.endswith('/'):
...
```

/e12/my_http_server.py

Question 2

What would happen if you remove the line ` socket.send_text_line(“”)`? Does removing the line introduce a bug? If so, what specifically goes wrong?

Code:

```
    if os.path.isfile(full_path):
        file_size = os.path.getsize(full_path) # Grab the file size
        _, file_extension = os.path.splitext(full_path) # Grab the extension from the path
        content_type = EXTENSION_MAP.get(file_extension) # Grab content type from our extensio
        with open(full_path, 'rb') as file:
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line(f"Content-Type: {content_type}")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("")

            socket.send_binary_data_from_file(file, file_size) # Send binary data since we are
...
```

/e13/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    #check file extension
    file_extension = os.path.splitext(path)[1].lower()

    #get content type based on files extension
    content_type = EXTENSION_MAP.get(file_extension, "text/plain")


    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
...
```

/e13/my_http_server.py

Question 2

What would happen if you remove the line ` socket.send_text_line(“”)`? Does removing the line introduce a bug? If so, what specifically goes wrong?

Code:

```
    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):

        # We need to know the file size so we can send
        # the Content-Length header.
        file_size = os.path.getsize(path)
        with open(path, 'rb') as file:
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line(f"Content-Type: {content_type}")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("") # <=====

            # Read and send one line at a time.
            # (This works because this server only handles text.)
            socket.send_binary_data_from_file(file, file_size)
...
```

/e14/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):

    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"RRRRRRRRRRRRRRRRequest: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts=request.split()
    path=parts[1][1:] #Grab second fragment of HTTP line    GET /[image.jpg] HTTP1.1
    print(path,"\n")
    #GET FILE EXTENSION, default is "text/plain"
    file_extension=os.path.splitext(path)[1]  #get  [] from image[.jpg]
    type1 = EXTENSION_MAP.get(file_extension, "text/plain")
    if os.path.exists(path):    # If the file name exists, we will send it as a response.
        file_size = os.path.getsize(path)

        # zk There is no need to handle text and binary separately.
        # Just treat them all as binary.
        if type1.split('/',1)[0] =="image" or type1 =="application/pdf":
...
```

/e14/my_http_server.py

Question 2

What would happen if you remove the line ` socket.send_text_line("")`? Does removing the line introduce a bug? If so, what specifically goes wrong?

Code:

```
        if type1.split('/',1)[0] =="image" or type1 =="application/pdf":
             print("type:",type1.split('/',1)[0])#......................
             #handle bunary
             with open(path, 'rb') as file:
                 socket.send_text_line("HTTP/1.0 200 OK")
                 socket.send_text_line(f"Content-Type: {type1}")
                 socket.send_text_line(f"Content-Length: {file_size}")
                 socket.send_text_line("Connection: close")
                 socket.send_text_line("")
                 #for pdf/img
                 print("before image is sent\n")
                 socket.send_binary_data_from_file(file,file_size)
                 print("Image was sent?\n")
...
```

/e15/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```python
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
        handle_request(path, socket)
...
```

/e15/handlers.py

Question 1

What specifically would go wrong if you removed the empty string from this list?

Code:

```
DEFAULT_HEADERS = ["Connection: close", ""]
...
```

/e16/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()

    # redirect if directory does not end with '/'
    if os.path.isdir(os.path.join(STUDENT_DATA_DIR, parts[1].strip("/"))) and not parts[1].ends
        socket.send_text_line("HTTP/1.0 301 Moved Permanently")
        socket.send_text_line(f"Location: {parts[1]}/")
        socket.send_text_line("Content-Length: 0")
        socket.send_text_line("Connection: close")
        socket.send_text_line("")
        socket.close()
        return

    # directory pathing
    path = os.path.join(STUDENT_DATA_DIR, parts[1].lstrip("/"))
    if os.path.isdir(path):
        # if path has index, set the path to the index.
        index_path = os.path.join(path, "index.html")
        if os.path.isfile(index_path):
            path = index_path
...
```

/e16/my_http_server.py

Question 2

What would happen if you remove the line ` socket.send_text_line(“”)`? Does removing the line introduce a bug? If so, what specifically goes wrong?

Code:

```
   # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):
        _, ext = os.path.splitext(path) #Source: https://stackoverflow.com/questions/541390/ex
        content_type = EXTENSION_MAP.get(ext.lower())

        # We need to know the file size so we can send
        # the Content-Length header.
        file_size = os.path.getsize(path)
        with open(path, "rb") as file:
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line(f"Content-Type: {content_type}")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("") # <======
            socket.send_binary_data_from_file(file, file_size)
...
```

/e17/my_http_server.py

Question 1

Add code (or pseudo code) to determine whether the client sent a GET or POST request.

Code:

```
def handle_connection(connection):
    """
    Handle the "conversation" with a single client connection
    """
    socket = http_socket.HTTPSocket(connection)

    # Read and print the request (e.g. "GET / HTTP/1.0")
    request = socket.receive_text_line()
    print(f"Request: {request}")


    # Read and print the request headers
    while True:
        data = socket.receive_text_line().strip()
        if (not data) or (len(data) == 0):
                break
        print(data)
    print('=======')

    # Extract the path from the request.
    parts = request.split()
    path = parts[1][1:]  # remove the first character of the path

    # Assume the path is a file name.
    # If the file name exists, we will send it as a response.
    # Otherwise, send a 404.
    if os.path.exists(path):

...
```

/e17/my_http_server.py

Question 2

What would happen if you remove the line ` socket.send_text_line(“”)`? Does removing the line introduce a bug? If so, what specifically goes wrong?

Code:

```
    if os.path.exists(path):

        # We need to know the file size so we can send
        # the Content-Length header.
        file_size = os.path.getsize(path)
        with open(path, 'r') as file:
            socket.send_text_line("HTTP/1.0 200 OK")
            socket.send_text_line("Content-Type: text/plain")
            socket.send_text_line(f"Content-Length: {file_size}")
            socket.send_text_line(f"Connection: close")
            socket.send_text_line("")

            # Read and send one line at a time.
            # (This works because this server only handles text.)
            while line := file.readline():
                socket.send_text_line(line)
    else:
...
```