Ex No: 3 **Implementation of HTTP Client using TCP Sockets**

Aim:

To develop a robust HTTP web client using raw TCP sockets (without high-level HTTP libraries) that downloads web pages and supports Parsing status line and headers, Following one level of HTTP 301/302 redirect, Handling responses with Content-Length or chunked transfer encoding and Saving text or binary content to a file.

Algorithm:

Step 1: Start the program.
Step 2: Input host name and resource path (e.g. example.com, /).
Step 3: Create a TCP socket connection to the host on port 80.
Step 4: Construct the HTTP GET request with required headers:

GET /path HTTP/1.1
Host: example.com
Connection: close

Step 5: Send the request through the socket.
Step 6: Receive the response in chunks until the server closes the connection.
Step 7: Split the response into status line, headers and body.
Step 8: Parse the status line and headers.

- If the status code is 301 or 302, extract the Location header and repeat steps 3–7 once more (single redirect).
- If Content-Length is present, read exactly that many bytes for the body.
- If Transfer-Encoding: chunked, decode the chunked body.

Step 9: Save the response body to a file.

- If Content-Type indicates text, save as .html.
- If binary (e.g., image), save in binary mode.

Step 10: Display logs of parsed headers, redirect trace, and saved file location.
Step 11: Close the socket and end the program.

Program Code (Python):

```
import socket
import ssl

def fetch_url(host, path="/", use_https=False, redirect_allowed=True):
    port = 443 if use_https else 80
```

```
# Create socket connection
sock = socket.create_connection((host, port))
if use_https:
    context = ssl.create_default_context()
    sock = context.wrap_socket(sock, server_hostname=host)

# Send HTTP GET request
request = f"GET {path} HTTP/1.1\r\nHost: {host}\r\nConnection: close\r\n\r\n"
sock.sendall(request.encode())

# Receive response
response = b""
while True:
    data = sock.recv(4096)
    if not data:
        break
    response += data
sock.close()

# Split headers and body
header_part, body = response.split(b"\r\n\r\n", 1)
headers = header_part.decode(errors="replace").split("\r\n")
status_line = headers[0]
header_dict = {}
for h in headers[1:]:
    if ": " in h:
        key, value = h.split(": ", 1)
        header_dict[key.lower()] = value

print("\nStatus Line:", status_line)
print("Headers:", header_dict)

# Handle Redirect (301/302)
if ("301" in status_line or "302" in status_line) and redirect_allowed:
    location = header_dict.get("location")
    if location:
        print("Following redirect to:", location)
        if location.startswith("http://"):
            new_host = location.split("/")[2]
```

```
            new_path = "/" + "/".join(location.split("/")[3:])
            return fetch_url(new_host, new_path, use_https=False, redirect_allowed=False)
        elif location.startswith("https://"):
            new_host = location.split("/")[2]
            new_path = "/" + "/".join(location.split("/")[3:])
            return fetch_url(new_host, new_path, use_https=True, redirect_allowed=False)

# Handle Content-Length
if "content-length" in header_dict:
    length = int(header_dict["content-length"])
    body = body[:length]

# Handle chunked transfer encoding
elif header_dict.get("transfer-encoding") == "chunked":
    decoded = b""
    while body:
        chunk_size_str, body = body.split(b"\r\n", 1)
        chunk_size = int(chunk_size_str.strip(), 16)
        if chunk_size == 0:
            break
        decoded += body[:chunk_size]
        body = body[chunk_size+2:]  # skip chunk data + CRLF
    body = decoded

# Save file (text or binary)
content_type = header_dict.get("content-type", "")
if "text" in content_type or "html" in content_type:
    filename = "output.html"
    with open(filename, "w", encoding="utf-8", errors="replace") as f:
        f.write(body.decode(errors="replace"))
else:
    filename = "output.bin"
    with open(filename, "wb") as f:
        f.write(body)

print(f"Saved response as {filename}")
return filename
```

```
# 1. Normal site with Content-Length
fetch_url("example.com", "/", use_https=False)

# 2. Redirect test
fetch_url("google.com", "/", use_https=False)

# 3. Chunked transfer test
fetch_url("httpbin.org", "/stream/5", use_https=False)

# 4. Binary file test
fetch_url("httpbin.org", "/image/png", use_https=False)
```

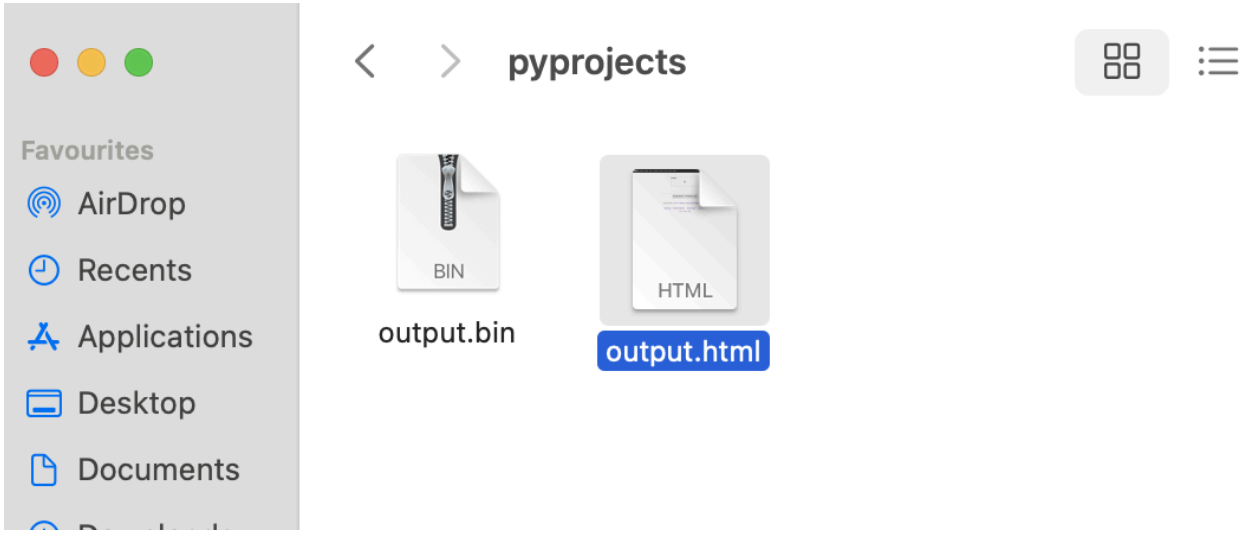
Result:

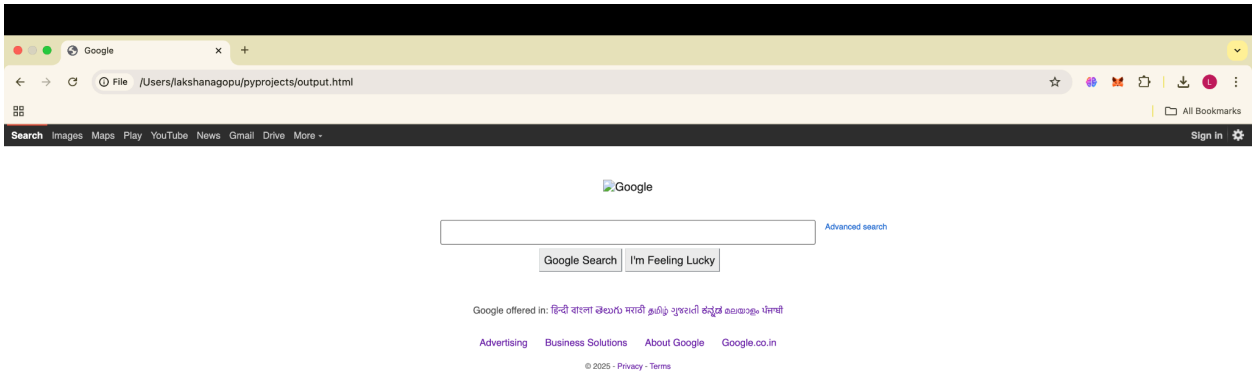
The HTTP client successfully Parsed the status line and headers, followed one level of HTTP 301/302 redirect, handled both Content-Length and chunked transfer encodings and saved both text and binary contents into appropriate files.

Output:

```
lakshanagopu@Lakshanas-MacBook-Air pyprojects % nano robustclient.py
lakshanagopu@Lakshanas-MacBook-Air pyprojects % python3 robustclient.py
Status Line: HTTP/1.1 200 OK
Headers: {'content-type': 'text/html', 'etag': '"84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"', 'last-modified': 'Mon, 13 Jan 20
25 20:11:20 GMT', 'cache-control': 'max-age=86000', 'date': 'Mon, 29 Sep 2025 08:12:33 GMT', 'content-length': '1256', 'connection': 'c
lose', 'x-n': 'S'}
Saved response as output.html
lakshanagopu@Lakshanas-MacBook-Air pyprojects % rm robustclient.py
lakshanagopu@Lakshanas-MacBook-Air pyprojects % nano robustclient.py
lakshanagopu@Lakshanas-MacBook-Air pyprojects % python3 robustclient.py

Status Line: HTTP/1.1 200 OK
Headers: {'content-type': 'text/html', 'etag': '"84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"', 'last-modified': 'Mon, 13 Jan 20
25 20:11:20 GMT', 'cache-control': 'max-age=86000', 'date': 'Mon, 29 Sep 2025 08:16:29 GMT', 'content-length': '1256', 'connection': 'c
lose', 'x-n': 'S'}
Saved response as output.html

Status Line: HTTP/1.1 301 Moved Permanently
Headers: {'location': 'http://www.google.com/', 'content-type': 'text/html; charset=UTF-8', 'content-security-policy-report-only': "obj
ect-src 'none';base-uri 'self';script-src 'nonce-SlN4BN3u6bWZo5PIRoQIzA' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'
 https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp", 'date': 'Mon, 29 Sep 2025 08:16:30 GMT', 'expires': 'Wed, 29 Oct
 2025 08:16:30 GMT', 'cache-control': 'public, max-age=2592000', 'server': 'gws', 'content-length': '219', 'x-xss-protection': '0', 'x-
frame-options': 'SAMEORIGIN', 'connection': 'close'}
Following redirect to: http://www.google.com/

Status Line: HTTP/1.1 200 OK
Headers: {'date': 'Mon, 29 Sep 2025 08:16:31 GMT', 'expires': '-1', 'cache-control': 'private, max-age=0', 'content-type': 'text/html;
charset=ISO-8859-1', 'content-security-policy-report-only': "object-src 'none';base-uri 'self';script-src 'nonce-zVCknQMgFIQMTijVXvnODg
' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp",
'p3p': 'CP="This is not a P3P policy! See g.co/p3phelp for more info."', 'server': 'gws', 'x-xss-protection': '0', 'x-frame-options': '
SAMEORIGIN', 'set-cookie': 'NID=525=pIKgNPhBdMxbLp0UMb9t2JouFekShjkmeBGpQE4BHMMONHyQgHfWc3k-G8qlcOTV6CKuZ8u2tNNXPTV6M5HT-2qgH1I2L90JWSe
4Frh8I5Q7FvTKuTjcz0l0TJeF1c_0464plrgr5rDCtF8z0D8u6HzqFd1hnNjpYJFOQEP94xMlm-a8UAl4nprR8sI9KetBPu6tHvIFDgTMGVXnEVA; expires=Tue, 31-Mar-2
026 08:16:31 GMT; path=/; domain=.google.com; HttpOnly', 'accept-ranges': 'none', 'vary': 'Accept-Encoding', 'connection': 'close', 'tr
ansfer-encoding': 'chunked'}
Saved response as output.html

Status Line: HTTP/1.1 200 OK
Headers: {'date': 'Mon, 29 Sep 2025 08:16:34 GMT', 'content-type': 'application/json', 'transfer-encoding': 'chunked', 'connection': 'c
lose', 'server': 'gunicorn/19.9.0', 'access-control-allow-origin': '*', 'access-control-allow-credentials': 'true'}
Saved response as output.bin

Status Line: HTTP/1.1 200 OK
Headers: {'date': 'Mon, 29 Sep 2025 08:16:35 GMT', 'content-type': 'image/png', 'content-length': '8090', 'connection': 'close', 'serve
r': 'gunicorn/19.9.0', 'access-control-allow-origin': '*', 'access-control-allow-credentials': 'true'}
Saved response as output.bin
lakshanagopu@Lakshanas-MacBook-Air pyprojects %
```

Saved HTML file opened in a browser





File size verification for Content-Length

```
lakshanagopu@Lakshanas-MacBook-Air pyprojects % ls -l output.html

-rw-r--r--@ 1 lakshanagopu  staff  56313 Sep 29 13:46 output.html
lakshanagopu@Lakshanas-MacBook-Air pyprojects %
```

The initial server response indicated Content-Length = 1256 bytes. After following a 301 redirect, the client downloaded the final page, which is 56313 bytes, corresponding to the full content of the redirected URL. This confirms the client correctly handled redirects and downloaded the complete content.

Binary file verification

```
lakshanagopu@Lakshanas-MacBook-Air pyprojects % file output.bin

output.bin: PNG image data, 100 x 100, 8-bit/color RGB, non-interlaced
lakshanagopu@Lakshanas-MacBook-Air pyprojects %
```

Checksum verification

```
lakshanagopu@Lakshanas-MacBook-Air pyprojects % curl -o ref.png http://httpbin.org/image/png
md5 output.bin ref.png

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8090  100  8090    0     0   1619      0  0:00:04  0:00:04 --:--:--  1835
MD5 (output.bin) = 5cca6069f68fbf739fce37e0963f21e7
MD5 (ref.png) = 5cca6069f68fbf739fce37e0963f21e7
lakshanagopu@Lakshanas-MacBook-Air pyprojects %
```