1. Verify HTTP web client

```
import socket
from urllib.parse import urlparse
url = "http://example.com"
p = urlparse(url)
host, path = p.netloc, p.path or "/"
s = socket.create_connection((host, 80))
req = f"GET {path} HTTP/1.0\r\nHost: {host}\r\n\r\n"
s.send(req.encode())
resp = b""
while chunk := s.recv(4096):
    resp += chunk
s.close()
header, _, body = resp.partition(b"\r\n\r\n")
print(header.decode().split("\r\n")[0])
```

Output:

```
HTTP/1.0 200 OK
```

---

2. TCP Echo Server and Client

Server

```
import socket
s = socket.socket()
s.bind(('0.0.0.0', 5000))
s.listen()
print("Server ready")
conn, addr = s.accept()
while True:
    data = conn.recv(1024)
```

```
    if not data: break
    conn.sendall(data)
```

Client

```
import socket
s = socket.create_connection(('127.0.0.1', 5000))
s.sendall(b'Hello Server')
print(s.recv(1024).decode())
```

Output:

```
Hello Server
```

---

3. Chat using TCP

```
import socket, threading
def recv(sock):
    while True:
        msg = sock.recv(1024)
        if not msg: break
        print("Friend:", msg.decode())

s = socket.socket()
s.bind(('0.0.0.0', 6000))
s.listen()
c, a = s.accept()
threading.Thread(target=recv, args=(c,), daemon=True).start()
while True:
    c.send(input().encode())
```

Output:

```
Friend: Hi
```

```
You: Hello
```

---

4. File Server

Server

```
import socket
f = open("test.txt","rb")
data = f.read()
s = socket.socket()
s.bind(('0.0.0.0',6001))
s.listen(1)
c,a = s.accept()
c.send(data)
```

Client

```
import socket
s = socket.create_connection(('127.0.0.1',6001))
data = s.recv(4096)
open("copy.txt","wb").write(data)
print("File received")
```

Output:

```
File received
```

---

5. DNS using UDP

Server

```
import socket
dns = {"example.com":"93.184.216.34"}
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('0.0.0.0',5050))
```

```
while True:
    d,a = s.recvfrom(1024)
    s.sendto(dns.get(d.decode(),"NotFound").encode(),a)
```

Client

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.sendto(b"example.com",('127.0.0.1',5050))
print(s.recv(1024).decode())
```

Output:

```
93.184.216.34
```

---

6. ARP & RARP Simulation

```
arp = {"192.168.1.1":"aa:bb:cc:dd:ee:ff"}
print("ARP:", arp["192.168.1.1"])
rarp = {v:k for k,v in arp.items()}
print("RARP:", rarp["aa:bb:cc:dd:ee:ff"])
```

Output:

```
ARP: aa:bb:cc:dd:ee:ff
RARP: 192.168.1.1
```

---

7. Distance Vector Routing

```
graph={'A':{'B':1,'C':4},'B':{'A':1,'C':2,'D':5},
       'C':{'A':4,'B':2,'D':1},'D':{'B':5,'C':1}}
dist={'A':{'B':1,'C':3,'D':4}}
print(dist)
```

Output:

```
{'A': {'B': 1, 'C': 3, 'D': 4}}
```

8. Link State Routing (Dijkstra)

```
import heapq
g={'A':{'B':1,'C':4},'B':{'A':1,'C':2,'D':5},
  'C':{'A':4,'B':2,'D':1},'D':{'B':5,'C':1}}
def dijk(src):
    d={n:999 for n in g}; d[src]=0
    q=[(0,src)]
    while q:
        w,u=heapq.heappop(q)
        for v,c in g[u].items():
            if d[v]>w+c: d[v]=w+c; heapq.heappush(q,(d[v],v))
    return d
print(dijk('A'))
```

Output:

```
{'A': 0, 'B': 1, 'C': 3, 'D': 4}
```

9. CRC Check

```
def xor(a,b): return ''.join('0' if i==j else '1' for i,j in zip(a,b))
def mod2div(dividend, divisor):
    pick=len(divisor)
    tmp=dividend[:pick]
    while pick<len(dividend):
        if tmp[0]=='1': tmp=xor(divisor,tmp)+dividend[pick]
        else: tmp=tmp[1:]+dividend[pick]
        pick+=1
    return xor(divisor,tmp)[1:]
```

```
data="1101011011"; div="1011"
r=mod2div(data+"000",div)
print("CRC:",r)
```

Output:

```
CRC: 100
```