Exercise Description

Objective:

Build a protocol-translating proxy server. This proxy will act as a middleman, allowing a **TCP client** to communicate with a **UDP server**, which it normally couldn't do.

Scenario:

We have two existing applications:

1. `client.py` : A **TCP** client that generates random math problems (e.g., `5 + 10`), sends them over a TCP stream, and waits for a TCP response.
2. `server.py` : A **UDP** server that listens for math problems, calculates the result, and sends the answer back via UDP.

These two cannot talk to each other directly because they use different transport protocols. Your task is to build `tcp_udp_proxy.py` to sit in the middle and translate the communication.

Architecture:

The data flow will look like this:

```
[TCP Client] <---TCP Connection---> [Your Proxy] <---UDP Datagrams---> [UDP Server]
```

Your Proxy's Responsibilities:

1. Listen for and accept incoming TCP connections from `client.py`.
2. When it receives data from the TCP client, it must forward that data as a UDP datagram to `server.py`.
3. When it receives a UDP response from the server, it must send that data back to the correct TCP client over its established connection.
4. Handle multiple clients connecting at once using the `select` module.

Your Task:

Complete the `tcp_udp_proxy_exercise.py` file by filling in the code marked with `TODO` comments. The client and server files are already complete and should not be modified.

Instructions for Running

You will need **three separate terminal windows** for this exercise. The order is important!

1. Terminal 1: Start the UDP Calculator Server

This server needs to be running first to receive requests.

```
python server.py localhost 10001
```

2. Terminal 2: Start Your Proxy Server

The proxy needs to listen on one port (e.g., 9000) and know the address of the UDP server (localhost 10001).

```
python tcp_udp_proxy_exercise.py localhost 9000 localhost 10001
```

If your proxy starts correctly, it won't print anything yet. It's just waiting.

3. Terminal 3: Start the TCP Client

The client connects to the *proxy*, not the UDP server.

```
python client.py localhost 9000
```

Expected Output:

- **Client Terminal:** Will show problems being sent and results being received.
- **Proxy Terminal:** Will print messages when a client connects and when they disconnect.
- **Server Terminal:** You will not see any output here, as the UDP server does its work silently.