Purpose: To use IP version 6.

You will build an upgraded TCP time client/server pair from your homework so that they use IP version 6.

Actually, nothing in *your* client/server has anything to do with the version of IP that your socket uses. In fact the only pieces of Comer's code that depend on the version of IP are his `connectsock` and `passivesock`.

Modify Comer's `connectsock` and `passivesock` so they make an IPv6 connection between two machines. You do not need to handle local host.

You have three ways to do this.
1) make copies of Comer's code to `connectsock.c.back` and `passivesock.c.bak` so you don't destroy the originals.
2) Do the whole homework in a special subdirectory so you are using copies of everything and use the "submit" program to submit.
3) modify Comer's code without makeing copies since this is the last project, but this works only if you've finished all the other projects.

Use your TCP client and server from homework 8 to test your modifications to Comer's code. No modifications to this client/server code are necessary.

Testing: `puma` and `cougar` have been set up to have dual IP stacks. No others have been setup. So you must run your client on one of these machines and your server on the other.

Instructor Testing: I'll be testing in a special subdirectory with my copies of the time server and client.

Handling localhost (something you do not need to do): The easiest way to handle this is to use the fact that if the "node" (i.e., hostname) is the `NULL` pointer, `getaddrinfo` will return the loop back address. The simplest way to do this is to go into the main program and modify Comer's switch so that it says:
`char* host=NULL;`
instead of:
`char* host="localhost";`
The more complicated way is to modify `connectsock` to do a `strcmp` on `host` and use `NULL` if it is equal to `"localhost"`. The most complicated code would be to modify connect so it tries IPv6 first and use IPv4 if it is not available. Servers are allowed to bind both IPv4 and IPv6 to the same port so a 4/6 server could be built.