On my program, each peer does the following

1. Get a list of other peers from the *hostsfile*.txt
2. Configure an IP4 address using *getaddrinfo* with hints and port 8888
3. Create an unbound socket with the above address
4. Bind the socket to the created address
5. Create a listener thread to listen for messages from the other peers
    a. This thread runs in a loop waiting for "ALIVE?" messages from other peers and when it gets one, it sends an "ACK!" message to acknowledge that it's alive to the other peers.
    b. When it has heard from all the other peers(be it alive or ack messages), it marks a shared variable(*ready*) as true and prints *READY* to standard error.
6. The main thread then sleep for 5 seconds to let other peers also finish the set-ups above
7. It then starts sending "ALIVE?" messages to all of the other peers up to 10 times with a delay of 2 seconds till it hears from all of them. The main thread knows if it has to stop sending by reading from an atomic boolean variable, *ready*, that the receiver thread updates thread-safely.

The Dockerfile is used to build the docker image with the latest ubuntu version. And the docker compose spins all of the programs(5) at the same time and each program prints *READY* to standard error when it's done.

Problems faced when developing the program

1. The first challenge I faced during this homework was not knowing exactly what the hostname return from *getnameinfo* is. My program was trying to find a peer in the host list solely by their hostname obtained from *getnameinfo*. For example looking for *peer1* when a message is received from peer1. This was making my program exit without being 'ready'. I later found out the names of programs is not just their name, but also includes the project name and network name. So, for peer1 it is *peer1.lab1_mynetwork*, instead of *peer1*. Although a naive mistake, I spent a few hours trying to debug this with lots of logging before figuring out and taking care of with string operations.
2. Another challenge I faced is that some of the peers' receiving threads kept exiting before the other peers heard back from them. This made the other peers to retry and eventually exit without the program being ready as well. The problems still persist though. After attending the TA office hours, I kept the receiving thread running in a loop continuously and doing that gave the peers time to hear back from everyone else.