Environment Setting

we provide a Dockerfile, and you must run it by following the instructions below.

0. install Docker (if not already installed)

If Docker is not installed on your local machine, you must install it first:

> Install Docker Desktop

Verify the installation with:

```
docker --version
```

1. Build the docker image

The Dockerfile defines the base environment for compiling and running your program.

Dockerfile:

```
FROM ubuntu@sha256:9cbed754112939e914291337b5e554b07ad7c392491dba6daf25eef1332a22e8

WORKDIR /workspace

RUN apt-get update && apt-get install -y \
    build-essential \
    gcc \
    make \
    iputils-ping

CMD ["/bin/bash"]
```

Build a Docker image `network-env` using the provided Dockerfile.

```
# Move to the directory where the Dockerfile is located
cd <path_to_dockerfile>

# For Mac
docker buildx build --platform=linux/amd64 -t network-env .

# For Window
docker build -t network-env .
```

2. Create a Docker network

Set up a custom bridge network `my-network` so containers can communicate with each other.

```
# Create a user-defined bridge network
docker network create --driver bridge my-network

# Verify the network
docker network ls
docker network inspect my-network
```

3. Create the server container

Start a container named `server` and attach it to the created network.

```
# For Mac
docker run --platform=linux/amd64 -it \
--name server \
--network my-network \
-v "$(pwd)":/workspace \
network-env

# For Window
docker run -it \
--name server \
--network my-network \
-v %cd%:/workspace \
network-env
```

4. Check the server container's IP address

Find the internal IP address of the `server` container to be used by the client.

```
# For Mac
docker inspect server | grep "IPAddress"

# For Window
docker inspect server | findstr "IPAddress"

# Example output:
"SecondaryIPAddresses": null,
"IPAddress": "",
"IPAddress": "172.18.0.2",
```

5. Create the client container

Start another contianer named `client` on the same network to connect to the server.

```
# For Mac
docker run --platform=linux/amd64 -it \
--name client \
--network my-network \
-v "$(pwd)":/workspace \
network-env

# For Window
docker run -it \
--name client \
--network my-network \
-v %cd%:/workspace \
network-env
```

6. Verify connectivity between containers

From inside the `client` container, use `ping` to confirm the `server` container is reachable.

```
ping 172.18.0.2

# Example (success):
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: icmp_seq=0 ttl=64 time=0.095 ms
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.085 ms
...

# Example (failure):
ping: connect: Network is unreachable
# or
ping: sendto: Host is unreachable
# or
Request timeout for icmp_seq 0
```

If you receive responses similar to the success case above, it means the server container is up and reachable.

7. Run your program

Finally, build and execute your program using the provided Makefile.

Your code must be implemented so that it can be built and run with these commands:

```
make all

# Server (Execute in server container)
./server -p 8888

# Client (Execute in client container)
./client -h 172.18.0.2 -p 8888 -o 0 -k unist < test.txt
```