Project 2: A Peer-to-Peer Messenger using TCP Sockets

Write a simple peer-to-peer messaging program (Messenger) in C/C++ using TCP sockets. You can accomplish this using the all of the message passing library (MPL) and Project 2 helper code discussed in class, but you're not required to use the helper code. You can write socket code directly if you choose to, but you have to meet the requirements specified below:

Requirements:

1. Your program (Messenger) should a display a menu prompt with at least two options as shown below:

   S – Send a message to Peer

   Enter Message: Hello, this is a message from peer 2 **<enter>**

   Enter IP Address: 127.0.0.1 <enter>

   Enter TCP Port: 5050 <enter>

   D – Display all messages from Peers

   1. Hello, this is a message from peer 2
   2. Hey there, this is a message from peer 3

   Q – Quit

2. Your Messenger should be able to support communication with multiple peers simultaneously (concurrently). Your output should include a test with at least 3 peers, and clearly demonstrate that the peers can each send and receive messages from each other.

That's it! Submit your code and screenshots to BB. Your code and screen shots should clearly demonstrate that you meet requirements 1 and 2 (above).

Important *Notes:*

a. ***Figure 2: "Instructor's Solution Output"*** *(below) provides a screen shot of the instructor's working MPeer solution as an example of acceptable functionality and output. Note: Your output does not need be to identical to that in Figure 2.*
b. *You can use all of the Project 2 helper code to complete your Project 2 submission: available here*: https://mwcorley79.github.io/MikeCorley/lecture23/Project2_helpers/Project2_helpers.tar.gz
c. *For those of you who would like to do more, you can! One suggestion is to extend the Messenger to support file transfer. I will give example code.*

Getting Started with Project 2 Helper Code

All that you need to do to complete Project 2 is to supply remaining the code required in Messenger.cpp. Specifically, you complete functions: *MessageHandler::AppProc()* and *bool SendMessage(const Message &message, const EndPoint &dstPeerEP)* That's it. (Note: you are not required to use this code. You can use as much of it as you wish. Here's how you get started:

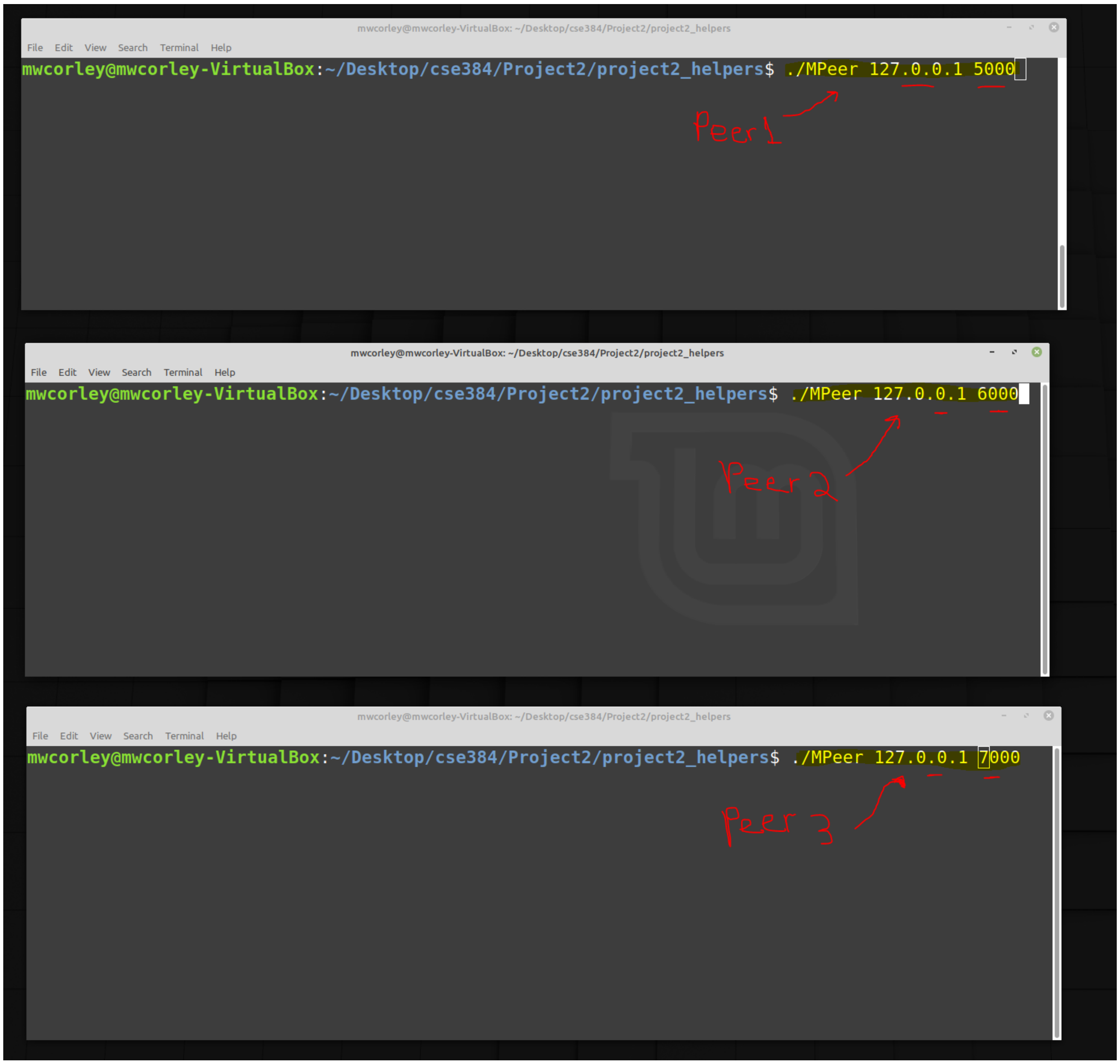
1. Download the Helper code from the class website here: https://mwcorley79.github.io/MikeCorley/lecture23/Project2_helpers/Project2_helpers.tar.gz

2. Unzip the tarball and run the following commands:
   cd project2_helpers
   ./project2_buildAll - this will build the socket message passing(static) library (libmpl.a) and the (Sender/Receiver) test stubs

3. Compile Messenger.cpp using the following command:
   g++ -o MPeer Messenger.cpp libmpl.a -lpthread - this will compile the Messenger.cpp and link against socket message passing library (libmpl.a), and the standard posix threads library (libpthread.a)

4. Run the Messenger using the following command:
   ./MPeer 127.0.0.1 5000

   127.0.0.1 – is localhost (the loopback IP address), and 5000 is an example TCP port used by the MPeer's listening server which is used to receive messages from other MPeers. You can choose any listening port you wish (above 1024).not being used by another process).
   For requirement 2 you need start a minimum of three MPeers. One suggestion is to start them all on localhost, on ports 5000, 6000, and 7000 as shown in ***Figure 1: "Running Multiple MPeers"*** (below)

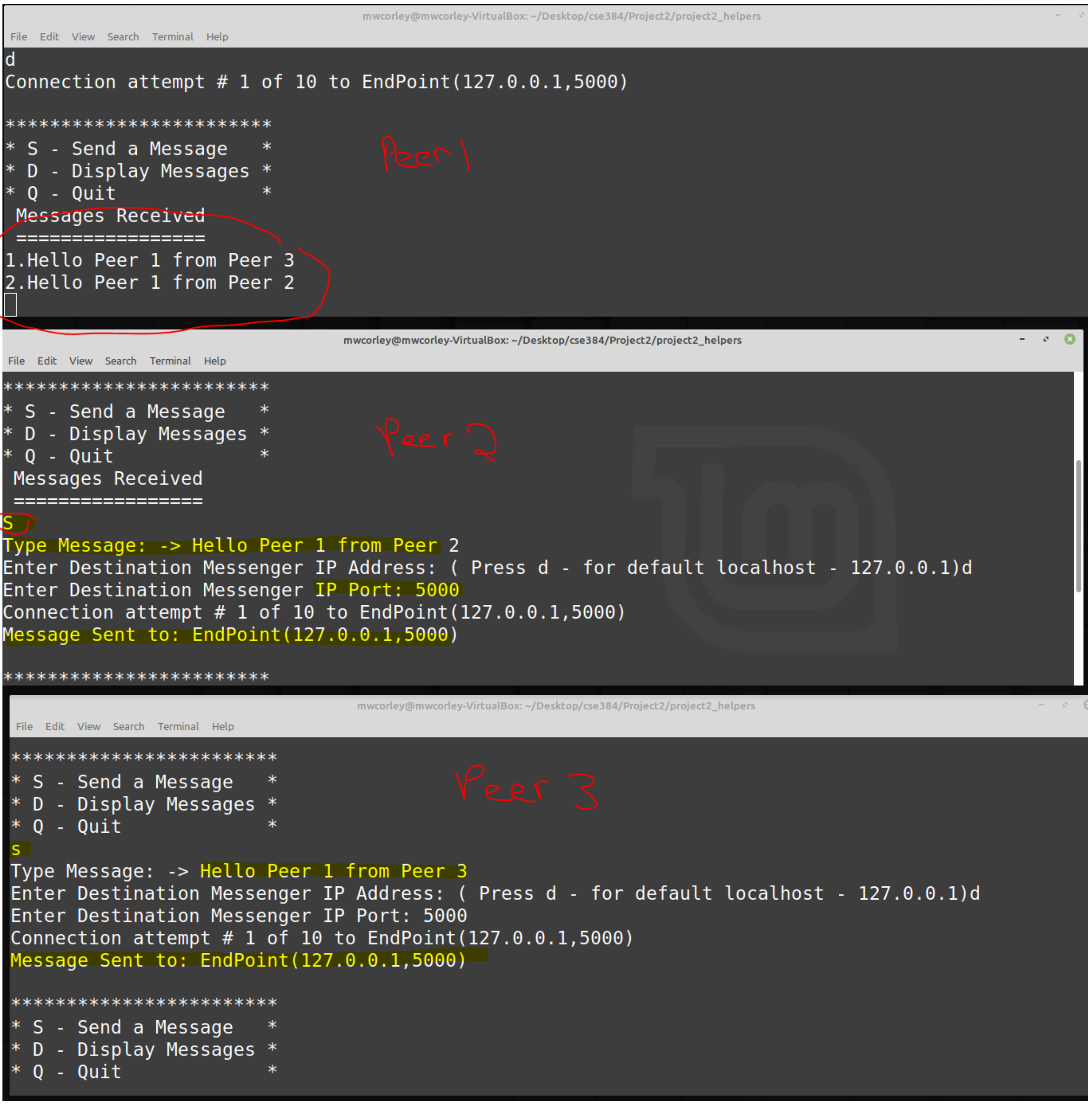


```
mwcorley@mwcorley-VirtualBox:~/Desktop/cse384/Project2/project2_helpers$ ./MPeer 127.0.0.1 5000
```

```
mwcorley@mwcorley-VirtualBox:~/Desktop/cse384/Project2/project2_helpers$ ./MPeer 127.0.0.1 6000
```

```
mwcorley@mwcorley-VirtualBox:~/Desktop/cse384/Project2/project2_helpers$ ./MPeer 127.0.0.1 7000
```

Figure 1: Running Multiple MPeers



mwcorley@mwcorley-VirtualBox: ~/Desktop/cse384/Project2/project2_helpers
File Edit View Search Terminal Help
Peer 1

```
d
Connection attempt # 1 of 10 to EndPoint(127.0.0.1,5000)

************************
* S - Send a Message    *
* D - Display Messages *
* Q - Quit              *
 Messages Received
 =================
1.Hello Peer 1 from Peer 3
2.Hello Peer 1 from Peer 2
```

mwcorley@mwcorley-VirtualBox: ~/Desktop/cse384/Project2/project2_helpers
File Edit View Search Terminal Help
Peer 2

```
************************
* S - Send a Message    *
* D - Display Messages *
* Q - Quit              *
 Messages Received
 =================
S
Type Message: -> Hello Peer 1 from Peer 2
Enter Destination Messenger IP Address: ( Press d - for default localhost - 127.0.0.1)d
Enter Destination Messenger IP Port: 5000
Connection attempt # 1 of 10 to EndPoint(127.0.0.1,5000)
Message Sent to: EndPoint(127.0.0.1,5000)

************************
```

mwcorley@mwcorley-VirtualBox: ~/Desktop/cse384/Project2/project2_helpers
File Edit View Search Terminal Help
Peer 3

```
************************
* S - Send a Message    *
* D - Display Messages *
* Q - Quit              *
s
Type Message: -> Hello Peer 1 from Peer 3
Enter Destination Messenger IP Address: ( Press d - for default localhost - 127.0.0.1)d
Enter Destination Messenger IP Port: 5000
Connection attempt # 1 of 10 to EndPoint(127.0.0.1,5000)
Message Sent to: EndPoint(127.0.0.1,5000)

************************
* S - Send a Message    *
* D - Display Messages *
* Q - Quit              *
```

Figure 2: Instructor's Solution Output