Question 3

Steps of Execution

1. Compile the program using: `gcc get_PID.cpp -o <name of object> -lstdc++`
2. Execute the program using: `sudo ./<name of object>`
3. The packet sniffer will now run for 30 seconds.
4. The program will output the set of ports available for query.
5. Enter `quit` to exit the program.
6. Do note that the number of PIDs and ports obtained can vary between different runs of the program. Try running it multiple times to get a suitable idea of the output.

Extracting the PIDs while running the code initially for 30 seconds

The working of the code in this question relies on the code and functions written in the first question, ie. sniffing packets and extracting data out of them. Building upon what was done before, for each packet we already have the source and destination ports for each packet. We can use the source port for each packet to determine the PID of the process that was using it. Our function of choice to determine the same is the "lsof" function. We can pass the additional parameter "-i" to get an extensive information about the running processes. In this extensive information about each process, we also have the port that each of the running processes is currently using. We then pipe this output to the grep output to extract the lines which contain the passed port. From each of these lines, we extract the 2nd column using the 'awk' command, which is the PID of the process of that row.

```
COMMAND   PID      USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
brave    4602 rachit0206   30u  IPv6 1398517      0t0  TCP Wannabe-Macbook:43798->[2600:9000:2553:1e00:12:bd7a:5b00:93a1]:https (ESTABLISHED)
brave    4602 rachit0206   34u  IPv4 1398964      0t0  UDP Wannabe-Macbook:39518->bom12s19-in-f14.1e100.net:https
brave    4602 rachit0206   35u  IPv4 1398965      0t0  UDP Wannabe-Macbook:53673->bom12s12-in-f10.1e100.net:https
brave    4602 rachit0206   36u  IPv4 1397349      0t0  TCP Wannabe-Macbook:60986->ec2-52-51-220-184.eu-west-1.compute.amazonaws.com:8282 (ESTABLISHED)
brave    4602 rachit0206   38u  IPv6 1408095      0t0  UDP Wannabe-Macbook:46501->bom12s18-in-x0a.1e100.net:https
brave    4602 rachit0206   41u  IPv6 1394312      0t0  TCP Wannabe-Macbook:55710->[2a04:4e42:24::649]:https (ESTABLISHED)
brave    4602 rachit0206   42u  IPv4 1394402      0t0  UDP Wannabe-Macbook:46022->bom12s12-in-f10.1e100.net:https
brave    4602 rachit0206   48u  IPv6 1399544      0t0  TCP Wannabe-Macbook:43792->[2600:9000:2553:1e00:12:bd7a:5b00:93a1]:https (ESTABLISHED)
brave    4602 rachit0206   50u  IPv6 1405235      0t0  UDP Wannabe-Macbook:33484->bom12s13-in-x0a.1e100.net:https
brave    4602 rachit0206   51u  IPv4 1399289      0t0  UDP Wannabe-Macbook:58010->bom12s19-in-f14.1e100.net:https
brave    4602 rachit0206   52u  IPv6 1402396      0t0  TCP Wannabe-Macbook:34342->bom07s29-in-x05.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   53u  IPv4 1400536      0t0  TCP Wannabe-Macbook:48492->bom07s30-in-f14.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   54u  IPv4 1400537      0t0  TCP Wannabe-Macbook:48494->bom07s30-in-f14.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   58u  IPv6 1399477      0t0  TCP Wannabe-Macbook:58414->bom07s32-in-x05.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   60u  IPv6 1402377      0t0  UDP Wannabe-Macbook:38325->bom12s17-in-x03.1e100.net:https
brave    4602 rachit0206   63u  IPv6 1394342      0t0  TCP Wannabe-Macbook:47976->bom07s31-in-x0e.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   65u  IPv6 1405221      0t0  UDP Wannabe-Macbook:51925->bom12s13-in-x0a.1e100.net:https
brave    4602 rachit0206   69u  IPv6 1405957      0t0  UDP Wannabe-Macbook:59065->bom12s13-in-x0a.1e100.net:https
brave    4602 rachit0206   70u  IPv6 1405185      0t0  TCP Wannabe-Macbook:58416->bom07s32-in-x05.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   71u  IPv6 1398497      0t0  TCP Wannabe-Macbook:37834->bom12s13-in-x01.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   74u  IPv6 1394346      0t0  TCP Wannabe-Macbook:47844->bom12s14-in-x0e.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   75u  IPv6 1407012      0t0  TCP Wannabe-Macbook:58306->bom07s30-in-x0e.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   76u  IPv6 1402409      0t0  TCP Wannabe-Macbook:37844->bom12s13-in-x01.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   81u  IPv6 1398496      0t0  TCP Wannabe-Macbook:37826->bom12s13-in-x01.1e100.net:https (ESTABLISHED)
brave    4602 rachit0206   88u  IPv4 1398501      0t0  UDP Wannabe-Macbook:34576->bom12s12-in-f10.1e100.net:https
brave    4602 rachit0206  101u  IPv6 1405219      0t0  UDP Wannabe-Macbook:58499->bom12s17-in-x03.1e100.net:https
firefox 10351 rachit0206   33u  IPv4 1398981      0t0  TCP Wannabe-Macbook:51990->55.65.117.34.bc.googleusercontent.com:https (ESTABLISHED)
```

Fig. 1 A sample output from the lsof -i command. As is visible, the 2nd columns contains the PID of the running process.

We use the 'popen' function that initiates a child process that does what we described before and saves the output to a buffer. The fgets function extracts the PID from the buffer.

Do note that the process of recognizing the PID of each process from the port needs to be done in ‘real’ time, ie. The recognition process needs to be done as soon as we get to port. This is because the ports the processes use might not necessarily be invariant with time.

We construct a map with ports as keys and PIDs as outputs to run the future loop.

Starting the query process

The sniffing is designed to be run for 30 seconds. After that, the program outputs a list of ports for which it was able to capture the corresponding process. An infinite query loop then follows, where the user can prompt the program with a port, the PID’s of the processes that used that port in the 30 seconds run previously.