Department of Computer Science
Computer Networks
Due: Thursday 25th Sept (21.00)

Your name:
TA Name:
Estimated Time: 21 hours

This is pair assignment, you may work either on your own or with a partner of your choice.

This assignment should be completed using C/C++, the hand-in format is up to you as long as the program compiles with the make command from the source folder.

Please use zip to bundle your source code and program submission. Do **NOT** include any hidden files (.git, .DS_Store .vscode) files in your submission. All code used to complete the assignment should be submitted, with a Makefile that can be used to compile your code and a README text file explaining how to compile and run your program(s) and possibly which hard-coded options can be changed where.

This assignment requires that you use your laptop to create a port scanning/knocking program that interacts with a server on 130.208.246.98.

Question:	Port Scanner	Puzzle Solving	Code Quality	Bonus	Total
Points:	35	55	10	10	110
Score:					

Speak easy to the port, and perhaps it will let you in.

In this assignment you will be introduced to the delights of packet crafting, bit twiddling and UDP subterfuge.

Somewhere on the TSAM server (130.208.246.98), a server is listening to some UDP ports in the range 4000-4100. Find the ports, send them the right packets, and use the secret knock to gain access to the secret information!

1. **Port Scanner** .... *35 points*
   Write a UDP port scanner, that takes in as arguments the IP address of the machine, and a range of ports to scan between. The scanner should be run with the command:

   ```
   ./scanner <IP address> <low port> <high port>
   ```

   Use it to scan between ports 4000-4100 on 130.208.246.98 and print out the open ports that you find in this range.

   This requires to send some UDP datagram to each of the ports and wait some limited time for a response.

   Do not rely on the ports always being the same. Also, note that UDP is an unreliable protocol. Some packets may be dropped randomly.

2. **Puzzle Solving** .... *55 points*
   You should have discovered 4 open ports in part 1. The ports you discovered are puzzle ports, safeguarding information about two additional ports (the secret ports) which are not showing up on your scan. Your task is to write ***a separate program*** to solve the puzzle ports, in order to reveal the two secret ports, a secret phrase and finally use this information to knock on the final port. Each port will send you instructions on how to reveal its secret if you send it a UDP message that is at least 6 characters long.

   The program should be run with the command:

   ```
   ./puzzlesolver <IP address> <port1> <port2> <port3> <port4>
   ```

   The program should interact with the ports discovered in part 1 by sending them a UDP message and then following the instructions provided by the puzzle ports.

   Notes and Hints:

   - The puzzle ports and their order will change over time, so do not hard code the ports, but rather supply them as command line arguments to your program.
   - The program must not make any assumptions about the order of the arguments (e.g., you can not assume the first argument to always be a the same puzzle port), but rather detect which port is which from the responses you get to some default message.
   - Any information that you get as a response from the puzzle ports can change over time and must not be hard-coded! Rather, your program must extract the information from the responses and store it in some variables to use later.
   - Most of the puzzles can be solved by using a normal UDP socket, only one needs a raw socket.

Points are distributed for solving:

(a) (10 points) S.E.C.R.E.T. port

(b) (15 points) Evil port

(c) (15 points) Checksum port

(d) (15 points) E.X.P.S.T.N. + Port knocking

3. **Code Quality** .... *10 points*

Points will be awarded for code quality, commenting and submission as follows:

(a) (3 points) Code compiles using the supplied Makefile.

(b) (2 points) Code follows command line invocations described above.

(c) (5 points) Code is well commented and well structured.

4. **Bonus** .... *10 points*

For 10 bonus points. After completing the port-knocking, you were sent a secret message, follow the instructions in the secret message for 10 bonus points.