Computer Networks 2021 Exercises - Unit 2

FAN: hang0013

NOTE: Each student's work unit is unique. You *must* use the work that has been generated for your FAN. If you do not, then you will fail this work unit.

NOTE: You must record your answers in the answer file EXACTLY as required, and commit and make sure your changes have been pushed to the github server, as they will otherwise not be counted.

NOTE: The topic coordinator will periodically run the automatic marking script, which will cause a file called unit2-results.pdf to be updated in your repository. You should check this file to make sure that your answers have been correctly counted. That file will contain the time and date that the marking script was last run, so that you can work out if it has been run since you last changed your answers. You are free to update your answers as often as you wish, until the deadline for the particular work unit.

1 Socket Programming General Knowledge

For each question, you must record your answer in the unit2-answers.txt *file in your git repository. Each statement is either true or false. You must record 't' if you think the statement is true, or 'f', if you think that the statement is false. Your answer must be lower case. Uppercase answers will be marked incorrect. For example, if you believed that the answer to the following question was potato, you would put the word potato at the end of the* rj= *line in the file* unit2-answers.txt.

Question#	Description
rj	*The potato is a white-flesh starchy vegetables from which hot chips are made*

The entry in unit2-answers.txt *would thus look like:*

```
# Question 'rj': The potato is a white-flesh starchy vegetables from which hot chips are made
```

```
rj=t
```

Templates for each answer are provided in `unit2-answers.txt` for your convenience.

Are the following statements true or false?

Question#	Statement
ab	The write() function for the C programming language can result in a 'broken pipe' condition.

Question#	Statement
ac	The socket() function for the C programming language sets the address of a network socket

Question#	Statement
ad	The socket() function for the C programming language creates a network socket

Question#	Statement
ae	The write() function for the C programming language can be used to write data to a socket.

Question#	Statement
af	The close() function for the C programming language can be called by either the initiator or receiver of a socket connection.

Question#	Statement
ag	The listen() function for the C programming language causes a socket to actively connect via the network to listen for applications that require it.

Question#	Statement
ah	The listen() function for the C programming language marks a socket as 'passive', ready to accept new connections.

Question#	Statement
ai	The connect() function for the C programming language normally requires the network address and layer 4 address of the target.

Question#	Statement
aj	The accept() function for the C programming language changes a network socket that is waiting for network connections into an active connection.

Question#	Statement
ak	A network socket uniquely identifies a connection between two networked applications

Question#	Statement
al	The connect() function for the C programming language is used by a client to connect to a server

Question#	Statement
am	The recv() function for the C programming language can only be used to read from UDP sockets

Question#	Statement
an	The close() function for the C programming language closes a server's socket and all open connections created by that socket.

Question#	Statement
ao	The recv() function for the C programming language normally waits for data to arrive, if none is waiting.

Question#	Statement
ap	The write() function for the C programming language is used to reply to a read() request from a server.

2 Socket Program Design

For each question, you must record your answer in the unit2-answers.txt file in your git repository. You will be presented with several short socket-based programmes written using various programming languages. These programmes have been scrambled, and you must unscramble them, by placing the statements in the correct order. Your answers will be the numbers of the lines, once they have been ordered correctly.

(Note that leading white space and comments are removed from the lines of the programmes. The programmes will be written in either Python, C or JavaScript.)

For example, you would answer the following question:

Line#	Text
1	Remove cake from the oven.
2	Collect the ingredients.
3	Put cake mix into the oven.
4	Mix the ingredients together.

Question#	Text
gh	First line.
gi	Second line.
gj	Third line.
gk	Fourth line.

By entering the following into your unit2-answers.txt file:

```
# Question 'gh': Place the lines of the supplied programme in the correct order.

gh=2

gi=4

gj=3
```

Templates for each answer are provided in `unit2-answers.txt` for your convenience.

Correct the order of the lines in the following simple network programme

Line#	Text
1	`client.destroy();`
2	`const client = new net.Socket();`
3	`}); // client.connect(...`
4	`client.on('data', (data) => {`
5	`client.write(`${process.argv[3]}\r\n`);`
6	`const net = require('net');`
7	`}); // client.on(...`
8	`client.connect({ port: 59898 }, process.argv[2], () => {`
9	`console.log(`Server says: ${data.toString('utf-8')}`);`

Question#	Text
aq	*First line.*
ar	*Second line.*
as	*Third line.*
at	*Fourth line.*
au	*Fifth line.*
av	*Sixth line.*
aw	*Seventh line.*
ax	*Eighth line.*
ay	*Ninth line.*

Correct the order of the lines in the following simple network programme

Line#	Text
1	`s.listen()`
2	`s.bind((HOST, PORT))`
3	`data = conn.recv(1024)`
4	`if not data:`
5	`conn, addr = s.accept()`
6	`with conn:`
7	`while True:`
8	`conn.sendall(data)`
9	`print('Connected by', addr)`
10	`import socket`
11	`break`
12	`with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:`

Question#	Text
az	*First line.*
ba	*Second line.*
bb	*Third line.*
bc	*Fourth line.*
bd	*Fifth line.*
be	*Sixth line.*
bf	*Seventh line.*
bg	*Eighth line.*
bh	*Ninth line.*
bi	*Tenth line.*
bj	*11th line.*
bk	*12th line.*

3 Socket Program Implementation

This question forms part of the DN/HD vs lower grade diagnosis. The pedagogical diagnosis is made based on the guidance from: https://www.flinders.edu.au/content/dam/documents/staff/policies/academic-students/grading-scheme.pdf. Specifically, in this item, the DN gate will be:

- iii. produced work which shows a developing capacity for original, critical and creative thinking over and above the essential requirements of the learning outcomes

and the HD gate will be:

- iii. consistently demonstrated knowledge skills and application at the highest level expected of a student at a given topic level

If you are running Windows, you will need to first install ncat from https://nmap.org/ncat/, and also NodeJS from nodejs.org

Write a simple network programme in JavaScript that listens on port 54321 and implements a simple game:

Tic-Tac-Toe. On receiving a connection, your server programme should send a blank tic-tac-toe

board as the following:

```
   |   |
---+---+---
   |   |
---+---+---
   |   |
```

It should read input from the client, and based on that input, progressively populate the board with O's and X's. The digit 1 corresponds to the top-left cell, and then working left-to-right and top-to-bottom, with 9 corresponding to the bottom-right cell.

The game does not need to implement any logic, just the updating and re-display of the board.

For example, if the client were to send 3 and then 5 and then 6, the server would send the following:

```
   |   | O
---+---+---
   |   |
---+---+---
   |   |
```

then,

```
   |   | O
---+---+---
   | X |
---+---+---
   |   |
```

and then,

```
   |   | O
---+---+---
   | X | O
---+---+---
   |   |
```

Finally, when the board has all 9 positions filled, then it will close the connection. Your solution should be placed in a single file, `unit2-tictactoe.js`, and committed to your github repository, and should be runable using a command line line:

```
node unit2-tictactoe.js
```

And you should be able to test it with a command like:

```
nc 127.0.0.1 54321
```