

Computer Networks 2021 Exercises - Unit 2

FAN: daym0019

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
```

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

Are the following statements true or false?

Question#	Statement
ab	The <code>send()</code> function for the C programming language performs the same function as <code>write()</code> , but without the possibility to set flags.

Question#	Statement
ac	A network socket usually contains addressing information for layer 3, but not layer 4

Question#	Statement
ad	The <code>send()</code> function for the C programming language can be used to write data to a connection-oriented socket.

Question#	Statement
ae	The <code>send()</code> function for the C programming language allows setting the destination address when writing to connection-less sockets.

Question#	Statement
af	The <code>connect()</code> function for the C programming language normally requires the network address, but not any layer 4 address of the target.

Question#	Statement
ag	The <code>send()</code> function for the C programming language can be used to write out-of-band data to a socket.

Question#	Statement
ah	The bind() function for the C programming language creates a network socket and binds it to a network address

Question#	Statement
ai	The accept() function for the C programming language checks for newly received socket connections.

Question#	Statement
aj	The socket() function for the C programming language requires a network address, to be able to create a socket.

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

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

Question#	Statement
an	The accept() function for the C programming language accepts all waiting network sockets each time it is called.

Question#	Statement
ao	The read() function for the C programming language can be used to receive data from sockets.

Question#	Statement
ap	The read() function for the C programming language is the best choice for reading data from a UDP socket.

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.
```

$$gh=2$$

$$gi=4$$

$$gj=3$$

$$gk=1$$

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	<code>console.log('Closed', socket.remoteAddress, 'port', socket.remotePort);</code>
2	<code>server.maxConnections = 20;</code>
3	<code>socket.on('data', (buffer) => {</code>
4	<code>}); // socket.on('data'...</code>
5	<code>const net = require('net');</code>
6	<code>console.log('Connection from', socket.remoteAddress, 'port', socket.remotePort);</code>
7	<code>}); // socket.on('end'...</code>
8	<code>console.log('Request from', socket.remoteAddress, 'port', socket.remotePort);</code>
9	<code>}); // server = net.createServer</code>
10	<code>socket.on('end', () => {</code>
11	<code>const server = net.createServer((socket) => {</code>
12	<code>server.listen(59898);</code>
13	<code>socket.write(`\${buffer.toString('utf-8').toUpperCase()}\n`);</code>

Question#	Text
aq	<i>First line.</i>
ar	<i>Second line.</i>
as	<i>Third line.</i>
at	<i>Fourth line.</i>
au	<i>Fifth line.</i>
av	<i>Sixth line.</i>
aw	<i>Seventh line.</i>
ax	<i>Eighth line.</i>
ay	<i>Ninth line.</i>
az	<i>Tenth line.</i>
ba	<i>11th line.</i>
bb	<i>12th line.</i>
bc	<i>13th line.</i>

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

Line#	Text
1	<code>}); // const server = ...</code>
2	<code>socket.end(`\${new Date()}\n`);</code>
3	<code>server.listen(59090);</code>
4	<code>const server = net.createServer((socket) => {</code>
5	<code>const net = require('net');</code>

Question#	Text
bd	<i>First line.</i>
be	<i>Second line.</i>
bf	<i>Third line.</i>
bg	<i>Fourth line.</i>
bh	<i>Fifth line.</i>

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:

Maze Walker. On receiving a connection, your server programme should send back the a simple maze display, prompting the user to move through the maze using w,a,s and d to move, where w is forwards, s is backwards and a and d are left and right, respectively. To make your life easier, the maze should always be the following, with the starting position of the player indicated by the X:

```
### #
#  #
# ###
```



```
#  #
```

```
#X###
```

It should read input from the client, and based on that input, move the X through the maze. The X may not be moved onto a # symbol, or moved outside of the maze. When the X is placed in the space in the top line of the maze, the connection should be closed.

The game does not need to implement any other logic.

For example, if the client were to send w, a, and then d, the server would send the following:

```
### #
```

```
#  #
```

```
# ###
```

```
#X  #
```

```
# ###
```

then,

```
### #
```

```
#  #
```

```
# ###
```

```
#X  #
```

```
# ###
```

and then,

```
### #
```

```
#  #
```

```
# ###
```

```
# X #
```

```
# ###
```

Your solution should be placed in a single file, `unit2-maze.js`, and should be runnable using a command line line:

```
node unit2-maze.js
```

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

nc 127.0.0.1 54321