Computer Networks 2021 Exercises - Unit 2

FAN: mcgh0008

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 t' if you think the statement is true, or ' t 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 connect() function for the C programming language is used by a server to connect to a client

Question#	Statement
`ac`	The recv() function for the C programming language is less flexible than the recvfrom() function.

Question#	Statement
`ad`	The send() function for the C programming language performs the same function as write(), but without the possibility to set flags.

Question#	Statement
`ae`	Socket programming refers to programming using the network sockets software abstraction

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

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

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

Question#	Statement
ai	A network socket usually contains addressing information for layer 3 and layer 4

Question#	Statement
aj	The write() function for the C programming language is used to write the address information into a socket.

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

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

Question#	Statement
am	The bind() function for the C programming language connects a network address to a network address

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

Question#	Statement
ao	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
ap	The read() function for the C programming language can be used on both servers and clients.

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
0	`}); // client.connect(...`
1	`console.log('Connected to server');`
2	`client.connect(59898, process.argv[2], () => {`
3	`rl.on('line', (line) => {`
4	`const net = require('net');`
5	`const rl = readline.createInterface({ input: process.stdin });`
6	`client.on('data', (data) => {`
7	`console.log(data.toString('utf-8'));`
8	`const client = new net.Socket();`
9	`rl.on('close', () => {`
10	`}); // rl.on('close'...`
11	`client.write(`${line}\n`);`
12	`}); // client.on('data'...`
13	`const readline = require('readline');`
14	`client.end();`
15	`}); // rl.on('line'...`

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.*
az	*Tenth line.*
ba	*11th line.*
bb	*12th line.*
bc	*13th line.*
bd	*14th line.*
be	*15th line.*
bf	*16th line.*

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

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

Question#	Text
bg	*First line.*
bh	*Second line.*
bi	*Third line.*
bj	*Fourth line.*
bk	*Fifth line.*
bl	*Sixth line.*
bm	*Seventh line.*
bn	*Eighth line.*
bo	*Ninth 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:

Guess My Number. On receiving a connection, your server programme should send back the a simple display, prompting the user to guess a number between 1 and 100. The number to guessed will be '42' every time, to keep things simple for you. Thus it should show the following when it receives a connection:

```
Guess my number between 1 and 100
Your guess?
```

It should read input from the client, and based on that input, display 'My number is higher', 'My number is lower' or 'You guessed it!'. If the number is correctly guessed, then the connection should be closed.

The game does not need to implement any other logic.

For example, if the client were to send 10, then 90, and then 42, the server would send the following:

```
My number is higher
Guess my number between 1 and 100
Your guess?
```

then,

```
My number is lower
Guess my number between 1 and 100
Your guess?
```

and then,

```
You guessed it!
Guess my number between 1 and 100
Your guess?
```

Finally, when all letters have been guessed, it will close the connection. Your solution should be placed in a single file, t unit2-guessnumber.js, and should be runable using a command line line:

```
node unit2-guessnumber.js
```

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

```
nc 127.0.0.1 54321
```