Web Science
Quiz 1: March 1, 2022

Enter your answers directly into this document (with the exception of #2 and #3).
All answers should be In Your Own Words, using complete sentences with proper spelling and grammar.

Save this document as: answers.docx (or .odf or .pdf) (-5 if wrong name). For all questions other than #2 and #3, you will not receive any credit for answers not placed in this document.

When finished with the quiz, put everything you wrote (this document, all code, etc.) on GitHub into a branch in your lab repo named: quiz1 (-5 if submitted incorrectly). **Do not submit your node_modules folder! (-15 if you submitted the node_modules folder)**

1. **Short answers** (25 points): (Answer in complete sentences, explain your answers)

    a. (5) How can I determine the type of device that my page is being displayed on? Give two examples of why I might care.

You can guess what kind of device the page is being displayed on by using media queries for css where you apply certain styles based on something like the screen width, height, or resolution, or you can view the user agent of the computer viewing the page and form a conclusion through there.
One reason why this is useful is because some webpages would want to display a mobile version of their website where the layout is more touch-friendly.
Another reason could be that the website has responsive elements in it where depending on the screen width some elements would change position. For example, there could be a nav bar that is usually on the top left of the screen but when the screen changed to a certain size it moves to the bottom of the page.

    b. (5) What is a package-lock.json file? What is it used for? Is it required?

The package-lock.json is used to describe the exact development environment used for the node project. It is not required.

    c. (5) What is npm? How does it work? Why is it used?

npm is an acronym for node package manager and is used to install and publish node programs, run scripts, and also automatically resolve dependencies for node packages. It is entirely possible to develop a node application without npm but that would require the developer and possibly the user to manually install every single package and its dependencies while you can just type npm install instead and have npm do the heavy lifting for you.

d. (10) Describe **in detail** the sequence(s) of transaction(s) for a frontend to request data from some external entity via Node.

1. User inputs some information into a form, and presses a submit button.
2. This form produces a GET request that the API receives and parses.
3. The result of that parsing is reassembled into an API request for the external entity.
4. Node sends the API request to the external entity and then receives an answer.
5. Node sends the contents of the response to the frontend.
6. Frontend updates after receiving the response.

4. (20) Provide **two** different explanations of the code below. The first explanation should be a high-level explanation (no less than four complete sentences) outlining what this code does to someone who has no coding experience. The second explanation should be a *detailed* one explaining line-by-line what the code does. If there are any errors in the code, fix them.

```
var net = require('net');     // import the net library

var sockets==[];              // initialize sockets to an empty
array
          // should just be one equals sign
var s = net.Server(function(socket) {// create a variable that
is a function
   sockets.push(socket);      // push the socket parameter to
the sockets array

   socket.on('data', function(d) {    // when receiving data run
this code
      for(var i=0; i<sockets.length;i++) {    // iterate through
the indices of the socket array
         if (sockets[i]==socket) continue;    // if we are at
the index where we pushed the socket variable to the array we
skip to the next iteration of the loop
         sockets[i].write(d);     // we write to the output the
contents of the socket at index i
      }
   });
   socket.on('end', function() { // when data is done
transmitting we clean up the socket array
      var i=sockets.indexOf(socket); // find the index of the
socket variable we pushed to the sockets array
      sockets.splice(i,1);   // and splice the array so we keep
everything but the sockets variable
   });
});

s.listen(8080);  // listen on port 8080
```

This is the server file for a node application. It creates a web socket on port 8080 where it starts reading in data and then ends on certain keywords. While reading in the data, it will also store the data in a container and also send certain data to the receiving end. After it is done receiving the data, it will then clean up everything that it stored and prepare to receive data again.

2. (+5) What is the name of the RPI-developed chat protocol popular in the 1990s?
The 80/20 model