# CS 142 Final Examination

Winter Quarter 2017

You have 3 hours (180 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes; all other sources of information, including laptops, cell phones, etc. are prohibited.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

_____

(Signature)

**Solutions**

_____

(Print your name, legibly!)

_____

(SUID - stanford email account for grading database key)

| Problem | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | |
|---|---|---|---|---|---|---|---|---|---|
| Score | | | | | | | | | |
| Max | 12 | 12 | 10 | 10 | 12 | 12 | 14 | 12 | |
| | | | | | | | | | |
| Problem | #9 | #10 | #11 | #12 | #13 | #14 | #15 | #16 | Total |
| Score | | | | | | | | | |
| Max | 10 | 12 | 12 | 12 | 10 | 9 | 9 | 12 | 180 |

## Problem #1 (12 points)

```
 1: <html>
 2:   <head>
 3:     <title>CS142 Final Exam</title>
 4:     <meta charset="UTF-8">
 5:     <link rel="stylesheet" type="text/css" href="main.css" />
 6:     <script src="index.js"></script>
 7:   </head>
 8:   <body>
 9:     <div id="container">
10:       <img src="photo.png">
11:       <a href="http://cs142.stanford.edu">CS 142</a>
12:       <input class="photoInfo"></input>
13:     </div>
14:   <body>
15: </html>
```

Please list below which lines in the HTML file above will or could cause the browser to send an HTTP request. For each HTTP request state if the the requested data will be fetched **synchronously** (immediately with browser processing of the page suspended until the response comes in), **asynchronously** (immediately with browser processing of the page continuing before the response comes in), or **deferred** (request is generated some time after the page is rendered by the browser).

**Line 5: synchronous GET request for `main.css`**
**Line 6: synchronous GET request for `index.js`**
**Line 10: asynchronous GET request for `photo.png`**
**Line 11: deferred GET request for `http://cs142.stanford.edu` (when clicked)**

## Problem #2 (12 points)

You are designing an API for a restaurant's website. Write a sample API call for each CRUD operation on a single "order" while adhering to RESTful API design principles. An example for listing all orders has been provided below.

Hint: Your Endpoint may use `:id`, as you've done in the projects, to represent the unique ID for a resource.

| | **HTTP Method** | **Endpoint** |
|---|---|---|
| List | `GET` | `/order/list` |
| C | **_POST_____** | **__/order_____** |
| R | **_GET_____** | **__/order/:id_____** |
| U | **_PUT_____** | **__/order/:id____** |
| D | **_DELETE__** | **__/order/:id_____** |

## Problem #3 (10 points)

The same origin policy of browser isolation controls access so one website's JavaScript can not access another website's cookies or use XMLHttpRequest to access another website. Explain the loophole in browsers that allows Cross Site Request Forgery (CSRF) attacks where a website may generate valid HTTP requests to another website.

**Even under the same origin policy, HTTP requests generated by an attacker's HTML will have the cookies of the request destination attached by the browser. For backends that use session cookies to authenticate requests these requests will look valid. Requests can be generated by having an "a" tag that baits the user into clicking on it (GET request) or by generating a form submit (GET or POST requests).**

## Problem #4 (10 points)

Suppose you run a bitcoin exchange and your web application uses HTTPS requests to fetch all content. You hire an intern and instruct the person to add your company logo to every view. The intern figures that the logo is not sensitive so uses unencrypted HTTP requests to fetch the logo .png file from your server.  Describe the security problem this change causes.

**The problem was described in class as "mixed context".  The HTTP GET request for the `logo.png` file will have the cookies including the session cookie attached allowing a "man-in-the-middle" attacker to steal and use to forge valid-looking requests to the backend.**

## Problem #5 (12 points)

An Object Relational Mapping (ORM) system maps an object system onto a relational database. For each of the following parts of a relational database, what is the corresponding part in an object system.

a. relational table
b. table column
c. table row

Hint: Object systems typically have a class structure with inheritance containing objects that have properties/attributes and methods.

**a. relational table - A collection of objects of a specific class.**
**b. table column - Property/attribute of a class of objects.**
**c. table row - An object instance.**

## Problem #6 (12 points)

Assume you are given a correctly functioning database with several secondary indexes. You delete one of the secondary indexes. For each of the following effects the index deletion could have, state if the effect is either possible or impossible. Provide a justification for your answer. If possible, describe a scenario in which it would happen. If impossible, describe why.

1. The database continues to correctly function with performance unchanged.
2. The database continues to correctly function with performance increased.
3. The database continues to correctly function with performance decreased.
4. The database stops correctly functioning.

1. **A secondary index that is not used often can be deleted and leave overall performance unchanged.**
2. **A secondary index that has a high overhead from being updated when data values change compared to the benefits to lookup operations can be deleted and have performance increase.**
3. **A secondary index that eliminates much scanning of objects on lookup queries will see a performance decrease when delete.**
4. **A secondary index is an optimizations that avoids scanning of objects so deleting it should continue functioning correctly, albeit potentially much slower. The implications on performance for delecting unique keys could be so horrific to the point of having a plausible argument the system fails.**

## Problem #7 (14 points)

Assume you are given a MEAN stack web application like we developed in class. Indicate whether the following concepts pertain to the **browser** (frontend) side, the **server** (backend) side, or **both** and give a brief one-sentence justification:

a. Inserting objects into a database
b. Rendering HTML documents
c. Sending HTTP GET requests
d. Accepting TCP connections
e. Running JavaScript code
f. Sending HTML, CSS, and JavaScript files
g. Reacting to a mouse click

**a. Server: The database is running in the backend and the code that accesses the database runs in the web server tier which is also in the backend.**
**b. Browser: The browser contains the HTML layout engine that renders HTML on the screen.**
**c. Browser: When processing HTML, or using the XMLHttpRequest API, the browser generates HTTP GET requests that are sent to the server.**
**d. Server: The browser speaking HTTP establishes a TCP connection with the web server which accepts the connection.**
**e. Both: In a MEAN stack, we run JavaScript both in the browser (Angular.js) and in the web server (Node.js).**
**f. Server: A web server in MEAN will service HTML, CSS, and JavaScript files to the browser.**
**g. Browser: DOM event handling on mouse clicks happens in the browser.**

## Problem #8 (12 points)

1. Explain how JavaScript running in a Node.js webserver can read a file from disk and send it out over the network to a browser without all the bytes of the file being brought into JavaScript variables.

**The Node.js Buffer class allows programs to read file data into a Buffer instance that is stored in memory separate from the JavaScript heap. The Buffer can be sent out a network connection without being read into the heap and optimizes sharing with pointers.**

2. Node.js has a `fs.readFileSync` call in the `fs` module that allows the contents of a file to be directly returned from the `fs.readFileSync` call. Although this call can be convenient to use, it is strongly discouraged. Explain why.

**The `fs.readFileSync` execution can take a relatively long time if the file must be read from a magnetic disk. During that time the Node.js event loop will be waiting for the current function call to return and can't move on to another function, removing any chance of concurrency from overlapping computation with I/O.**

## Problem #9 (10 points)

The following Node.js program uses the Node `fs` module to read a large file twice using two different API calls. When run, the programs print the numbers 1 through 5 to the console. Answer this question by listing the order in which the numbers are printed. Provide a brief (one or two sentence) explanation of the order in your answer.

```
var fs = require("fs");

fs.readFile("./largeFile", function () {
  console.log("1");
});
console.log("2");

function readFileSyncWithCb(fileName, callback) {
  var f = fs.readFileSync(fileName);
  console.log("3");
  callback();
}

readFileSyncWithCb("./largeFile", function () {
  console.log("4");
});
console.log("5");
```

**Output:**

**2**
**3**
**4**
**5**
**1**

**The fs.readFile call will call its done callback later so the first log we get is '2'. Afterwards, readFileSyncWithCb will be called and print '3' and then call its callback which prints '4'. Execution then continues with '5' printed and later the fs.readFile callback will fire, printing '1';**

## Problem #10 (12 points)

The following Node.js program uses the Node `events` and `fs` modules to read an input file and print some lines to the console log.

```
var events = require('events');
var myEmitter = new events.EventEmitter();

var fs = require('fs');

myEmitter.on('A', function () {
  console.log('A');
});

myEmitter.on('D', function () {
  console.log('D');
});

var readableStreamEvent = fs.createReadStream('./inputFile');

readableStreamEvent.on('data', function (fileData) {
  console.log('B');
  myEmitter.emit('A');
});

myEmitter.emit('D');

readableStreamEvent.on('finish', function () {
  console.log('C');
  myEmitter.emit('A');
});
```

**Answer questions on the following page…..**

**… problem #10 continued from previous page.**

1. Assuming that `inputFile` contains exactly one character, list the letters that will be printed to the console log in the order in which they will be printed.

**D**
**B**
**A**
**C**
**A**

2. Assuming you have control over the contents of `inputFile`, is it possible to see A printed more than twice? Why or why not? Briefly justify your answer.

**The 'data' event may be emitted multiple times for successive chunks of the file, assuming the file is big enough. We just need to make the file big enough such that the stream returns the file contents in two or more chunks.**

3. Assuming you have control over the contents of inputFile, is it possible to see A printed less than twice? Why or why not? Briefly justify your answer.

**If the inputFile was empty, no 'data' events would be emitted so only the 'A' in the finish event would be printed.**

## Problem #11 (12 points)

1. Explain how a Content Distribution Network could greatly improve the performance on one web application yet not be beneficial for another web application. Give examples of each scenario in your answer.

**Content Distribution Networks work will for data that doesn't change (i.e. is read-only) and can be very beneficial but doesn't work at all for rapidly changing data. An application with mostly read-only data benefits whereas one with no read-only data will not.**

2. Explain the key difference between using a cloud computing service versus buying your own machines that makes cloud computing such a win for start up companies that hope to make it big.

**The pay for resource you use pricing model of Cloud providers is a big win for a start up that is using few resources but would like to rapidly expand resource consumption if the customers show up. Buying your own computing resources to be ready when the customers show can take much capital up front.**

## Problem #12 (12 points)

1. How is load balancing done in a scale-out storage system (e.g. a database system)? Explain why this can be harder than with scale-out web server architectures.

**Load balancing with web servers can be done by making every server able to process any requests so the request can assigned to web servers using any algorithm that distributed the request processing among the servers. When distributing load on a scale-out storage system, the requests need to be routed to the servers that contain the requested data. Balancing load means distributing the data so that requests are spread evenly among the data shards. Allocating data to servers to balance load can be more complex than load balancing when any stateless server can be used.**

2. Explain why many web applications backends have found it useful to have a fast but unreliable storage system to complement their reliable storage system.

**Application session state (state kept per browser connection) requires fast access so that different servers in a scale-out architecture can be load balanced across. Systems that tradeoff storing with lower reliability for lower resource consumption and higher speed, like memcached, are popular for session state.**

## Problem #13 (10 points)

Network protocols are constructed in layers with each layer only communicating with the layer directly below it and above it. A protocol is said to run on top of the layer that is directly below it. Order the following protocols to match the layering in real system. Each protocol (except the bottom most one) should be on top of the protocol that it runs on.

1. Internet Protocol (IP)
2. HyperText Transport Protocol (HTTP)
3. WiFi wireless radio
4. Transmission Control Protocol (TCP)
5. Representational state transfer (REST)

**REST**
**HTTP**
**TCP**
**IP**
**WiFi**

## Problem #14 (9 points)

Storing web application state using the HTTP cookie mechanism has some advantages including the ability to use the storage space on the user's machine. Assuming we can get over problems with cookies being lost or corrupted and browser imposed limits, what would be the problem with storing significant amounts of state in cookies?

**Since cookies are attached to every HTTP request sent from the browser to the website, having a lot of information stored in cookies would make every request bigger, consuming more bandwidth and slowing things down.**

## Problem #15 (9 points)

Describe an attack that can be launched if a hacker could become the Domain Name Service (DNS) server for a user's browser.

**If the hacker could substitute their own DNS service they could make any HTTP/HTTPS URL attach to the attacker's web server rather than where it was suppose to go. The attacker could host a fake website that fools the user into interacting with it.**

## Problem #16 (12 points)

In general any information coming in over the Internet to the backend of an web application must be treated as suspect by the backend. Even if the information is known to come from the web application's code running in a user's browser, an attacker might have hacked the browser to view or manipulate the application's data or communication to the backend.

Because of this lack of trust in the web frontend environment, having the web app frontend hold information for the backend can be tricky. For example, if the backend passes some information to the frontend the attacker in the browser could view, delete, or modify the information, or even create fake versions of the data.

1. Describe a mechanism covered in class that would allow the backend to send information to the frontend and when it came back the backend could tell if it had been modified or faked.

**By adding a message authentication code (MAC) to the information the backend and verify that the information returned has not been modified or faked..**

2. Describe a mechanism covered in class that would allow the backend to send information to the frontend and both detect if it was changed and be confident an attacker wasn't able to view it.

**By encrypting the information before adding a MAC the frontend can both detect if it was changed and be confident it wasn't viewed by someone.**