Quiz 1

out of 109 points

Print your name on the line below.

_____

Do not turn this page over until told by the staff to do so.

This quiz is "closed-book." However, you may utilize during the quiz one two-sided page (8.5" × 11") of notes, typed or written, and a pen or pencil, nothing else.

Scrap paper is included at this document's end.
Unless otherwise noted, you may call any functions we've encountered this term in code that you write.
You needn't comment code that you write, but comments may help in cases of partial credit.
If running short on time, you may resort to pseudocode for potential partial credit.

Circle your teaching fellow's name.

Aidi Zhang	Emily Houlihan	Kevin Mu	Rob Bowden
Alex Pong	Eric Ouyang	Lily Tsai	Robbie Gibson
Allison Buchholtz-Au	Frederick Widjaja	Luciano Arango	Saheela Ibraheem
Ankit Gupta	Gabriel Guimaraes	Luis Perez	Sam Green
Armaghan Behlum	Gal Koplewitz	Lukas Missik	Theo Levine
Arvind Narayanan	George Lok	Marcus Powers	Tiffany Wu
Belinda Zeng	Hannah Blumberg	Mehdi Aourir	Tim McLaughlin
Camille Rekhson	Ian Nightingale	Michael Patterson	Tomas Reimers
Chris Lim	Jackson Steinkamp	Michelle Danoff	Tony Ho
Cynthia Meng	Jason Hirschhorn	Nicholas Larus-Stone	Vipul Shekhawat
Dan Bradley	Jonathan Miller	Nick Joseph	Wellie Chao
Daven Farnham	Jordan Canedy	Nick Mahlangu	Wesley Chen
David Kaufman	Joshua Meier	Rei Otake	Willy Xiao
Doug Lloyd	Keenan Monks	Rhed Shi	Winnie Wu

for staff use only

final score out of 109

Stack Attack.

Consider the (global) declaration and initialization of a stack for integers, below, wherein `CAPACITY` is some constant.

```
struct
{
    int data[CAPACITY];
    int size;
}
stack;
stack.size = 0;
```

Note the absence of `typedef`. With the above, we're simply declaring one global `struct` called `stack`, whose `size` is initially `0`.

0. (4 points.) Complete the implementation of `push`, below, in such a way that the function pushes `datum` on top of `stack` if not already at capacity. Upon success, the function should return `true`. Upon failure (e.g., `stack` is at capacity), the function should return `false`.

```
bool push(int datum)
{
```

1. (4 points.) Complete the implementation of `pop`, below, in such a way that the function pops off the topmost integer from `stack`. Upon success, the function should store that integer at the address in `location` and then return `true`. Upon failure, the function should return `false`, without changing the value at the address in `location`. You may assume that `location` will be a valid non-`NULL` pointer.

```
bool pop(int* location)
{
```

for staff use only
points off
initials

Hash this out.

Consider the (global) definition below of a hash table for English words, each of whose characters is a letter of the English alphabet. Assume that each of the pointers in `table` will be initialized to `NULL`.

```
typedef struct node
{
    char* word;
    struct node* next;
}
node;

node* table[26];
```

2. (2 points.) Consider the hash function below, which assumes that `word` is a string of non-zero length, each of whose characters is a letter of the English alphabet.

```
int hash(const char* word)
{
    return (toupper(word[0]) - 'A');
}
```

While simple (and thus fast to compute), this hash function isn't necessarily the best. Critique this hash function, identifying and explaining a downside.

3. (4 points.) Suppose that `table` has been implemented without any sort of counter (i.e., variable) for keeping track of the number of words that have been inserted into `table`. And so the only way to determine the size of (i.e., number of words in) `table` is to traverse the whole structure. Complete the implementation of `size`, below, in such a way that the function determines and returns the number of words in `table`.

```
unsigned int size(void)
{
```

for staff use only
points off
initials

Trie this.

4. (4 points.) Consider the (global) definition below of a case-insensitive trie for English words, each of which is entirely alphabetical.

```
typedef struct node
{
    bool word;
    struct node* children[26];
}
node;

node* trie = NULL;
```

Suppose that `trie` has been implemented without any sort of counter (i.e., variable) for keeping track of the number of words in `trie`. And so the only way to determine the size of (i.e., number of words in) `trie` is to traverse the whole structure. Complete the implementation of `size`, below, in such a way that, when passed `trie`, the function returns the number of words in `trie`.

```
unsigned int size(node* n)
{
```

for staff use only
points off
initials

Grammatically Correct.

Recall that HTTP messages adhere to a "grammar," which is to say that they're formatted according to a set of rules, patterns that web servers and web browsers can parse. Consider the (simplified) grammar below, wherein any underlined symbol is further defined by some other rule. Know that `CRLF` represents `\r\n`, that `SP` represents a single white space, that `*` means "zero or more" (of whatever's in parentheses), that `/` means "or" and that square brackets mean something's optional.

```
HTTP-message    =  start-line
                   *( header-field CRLF )
                    CRLF
                    [ message-body ]
start-line      =  request-line / status-line
request-line    =  method SP request-target SP HTTP-version CRLF
request-target  =  absolute-path [ "?" query ]
status-line     =  HTTP-version SP status-code SP reason-phrase CRLF
header-field    =  field-name ":" field-value
```

5. (5 points.) Consider the HTTP message below.

```
GET /register.php?name=David&dorm=Matthews+Hall HTTP/1.1
Host: localhost
```

Parse the message by completing the table below, writing to the right of each symbol the substring to which it corresponds. We've parsed `start-line` for you.

`start-line`	`GET /register.php?name=David&dorm=Matthews+Hall HTTP/1.1`
`request-line`	
`method`	
`request-target`	
`HTTP-version`	
`absolute-path`	
`query`	
`header-field`	
`field-name`	
`field-value`	

for staff use only
points off
initials

6. (4 points.) Consider the HTTP message below.

```
HTTP/1.1 200 OK
Content-Type: application/json

{"symbol":"FREE","name":"FreeSeas Inc.","price":0.15}
```

Parse the message by completing the table below, writing to the right of each symbol the substring to which it corresponds. We've parsed `start-line` for you.

`start-line`	`HTTP/1.1 200 OK`
`status-line`	
`HTTP-version`	
`status-code`	
`reason-phrase`	
`header-field`	
`field-name`	
`field-value`	
`message-body`	

for staff use only
points off
initials

Better GET permission. (2 points each.)

Consider the hierarchical listing of directories and files below, between whose square brackets are the directories' and files' permissions.

```
[drwxr-xr-x]  /
`-- [drwxr-xr-x]  home
    `-- [drwx--x--x]  jharvard
        `-- [drwx--x--x]  vhosts
            `-- [drwx--x--x]  localhost
                `-- [drwx--x--x]  public
                    |-- [drwx------]  img
                    |   `-- [-rw-r--r--]  ajax-loader.gif
                    |-- [-rw-r--r--]  index.html
                    `-- [-rw-------]  quote.php
```

For each of the HTTP request lines below, state what the status code in the server's response would be and why it would be so. Assume that the web server's root is `/home/jharvard/vhosts/localhost/public`.

7. `GET / HTTP/1.1`

8. `GET /index.html HTTP/1.1`

9. `GET /quote.html HTTP/1.1`

10. `GET /img/ajax-loader.gif HTTP/1.1`

for staff use only
points off
initials

Delete cookies?!

Recall that a web server might respond to an HTTP request with lines like the below.

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=3vt2ut4vtr9mh2e3il83fcsln6; path=/
Content-Type: text/html
```

11. (2 points.) Exactly what is a cookie?

12. (2 points.) Explain the relationship between `PHPSESSID` and `$_SESSION`.

13. (2 points.) In the earliest days of online advertising, to "pixel" a site meant embedding a 1-pixel image (that's effectively invisible) from an advertiser's server (e.g., `example.com`) in each of a site's pages in order to keep track of the pages a user has visited. For instance, an advertiser might include on each page an HTML tag like the below.

    ```
    <img alt="" src="http://example.com/small.gif"/>
    ```

    Using your knowledge of HTTP, explain how such a tag might indeed enable an advertiser to track a user not only across one site's pages but across multiple sites that an advertiser has been allowed to pixel.

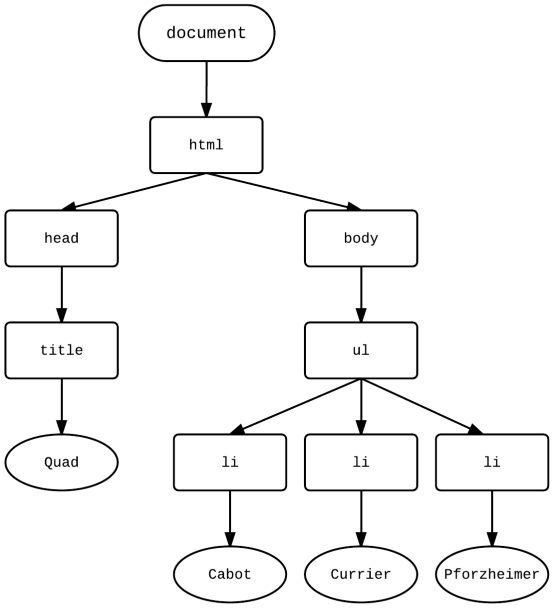
for staff use only
points off
initials

DOM, DOM DOM DOM.

14. (6 points.) Recall that an HTML document can be represented in memory with a DOM tree whose nodes represent the document's elements and text (and more). For instance, the HTML below at left can be represented with the DOM tree below at right, wherein rectangles represent elements (i.e., tags) and ovals represent text.

```
<!DOCTYPE html>

<html>
    <head>
        <title>Quad</title>
    </head>
    <body>
        <ul>
            <li>Cabot</li>
            <li>Currier</li>
            <li>Pforzheimer</li>
        </ul>
    </body>
</html>
```



Assume that, when implemented as a `struct` called `node` in C, a DOM node for elements has these fields: `childNodes`, via which the `node`'s children can be accessed; `nextSibling`, via which a `node`'s next sibling can be accessed; and `nodeName`, which is the name of a `node` (e.g., `"title"`). Complete the implementation of `node`, below, by specifying types for each of these fields. (You may add additional fields as well if needed.) Explain in comments why you have chosen the types that you have for each of your fields.

```
typedef struct node
{
```

for staff use only
points off
initials

Frosh IMs.

Consider the HTML below.

```
<!DOCTYPE html>

<html>
    <head>
        <script src="http://code.jquery.com/jquery-latest.min.js"></script>
        <script src="/scripts.js"></script>
        <title>Frosh IMs</title>
    </head>
    <body>
        <form action="/register.php" id="registration" method="get">
            <input id="name" name="name" placeholder="Name" type="text"/>
            <br/>
            <select id="dorm" name="dorm" size="1">
                <option value="">Dorm</option>
                <option value="Matthews Hall">Matthews Hall</option>
                <option value="Other">Other</option>
            </select>
            <br/>
            <input type="submit" value="Register"/>
        </form>
    </body>
</html>
```

15. (2 points.) Suppose that David from Matthews Hall submits this form. Complete the request line, below, of the HTTP request that the form's submission will generate.

```
GET
```

16. (4 points.) Complete the implementation of `scripts.js`, below, in such a way that it prevents the form's submission if a user fails to input a name or select a dorm.

```
$(function() {
```

for staff use only
points off
initials

Having said that...

Consider the remarks below, each of which sounds like an advantage but is not without an underlying disadvantage too. Complete each of the remarks, making clear the price paid (i.e., tradeoff) for the advantage.

17. (2 points.) Client-side validation of users' input into forms is generally faster than server-side validation, since you don't need to wait for a server's response. Having said that...

18. (2 points.) It often takes less time to implement a program in PHP than in C. Having said that...

19. (2 points.) A sorted array supports binary search whereas a sorted linked list does not. Having said that...

Why bother?

20. (2 points.) Why bother with hash tables with separate chaining? They're more complicated to implement than linked lists, and lookups are still in $O(n)$.

21. (2 points.) Why bother with external CSS files (e.g., `styles.css`) when you can just add a `style` attribute to HTML elements?

for staff use only
points off
initials

World Wide Wait.

22. (8 points.) Suppose that a user types `https://bankofamerica.com/` into his or her browser's address bar for the first time ever and then hits Enter. Explain, in a short paragraph, the process by which Bank of America's home page is retrieved, using each of the eight (8) terms below at least once in your explanation in such a way that your understanding of each is clear. Underline every instance of each term in your paragraph.

DNS · HTML · HTTP · HTTPS · IP address · port · router · TCP

for staff use only
points off
initials

Design Decisions.

For each pair below, *x* versus *y*, argue in no more than two sentences when you should use *x* rather than *y* (or, if you prefer, *y* rather than *x*).

23. (2 points.) `CHAR` versus `VARCHAR`

24. (2 points.) `DECIMAL` versus `FLOAT`

25. (2 points.) GET versus POST

26. (2 points.) `NULL` versus `'\0'`

27. (2 points.) CSV file versus SQL table

for staff use only
points off
initials

Hi, SQL.

28. (8 points.) Suppose that every student in Harvard's SQL database has a unique id, a unique email address, a name, and a house (i.e., dorm). Complete the schema below by providing a type, a length (if applicable), and an index (if appropriate) for each of the fields. Assume that this table shall be the only table in the database.

Name	Type	Length	Index
id			
email			
name			
house			

29. (4 points.) Suppose that President Skroob (whose id in a database is `6`) wants to add one (1) share of FreeSeas Inc. (whose symbol is `FREE`) to his portfolio. Consider the code for this transaction below, where `query` behaves as it did in Problem Set 7 and Problem Set 8.

```
$rows = query("SELECT shares FROM portfolios WHERE id = 6 AND symbol = 'FREE'");
if (count($rows) == 0)
{
    query("INSERT INTO portfolios (id, symbol, shares) VALUES(6, 'FREE', 1)");
}
else
{
    query("UPDATE portfolios SET shares = shares + 1 WHERE id = 6 AND symbol = 'FREE'");
}
```

Unfortunately, this code is vulnerable to a "race condition," much like a refrigerator that's run out of milk! Explain the bug in this code and propose how to fix.

for staff use only
points off
initials

Supercookie!

Consider the (simplified) depiction below of how Verizon's "supercookies" appear to work (courtesy of Jonathan Mayer at Stanford).

1. Device sends an HTTP request.

2. Verizon injects an HTTP header ("X-UIDH"). It's a temporary ID, hashed or HMACed with a key.

3. Destination website (or third party) receives HTTP request with injected header.

30. (2 points.) Even though Verizon isn't injecting a `Set-Cookie` or `Cookie` header into devices' HTTP requests, in what way is Verizon's `X-UIDH` header nonetheless similar to a cookie?

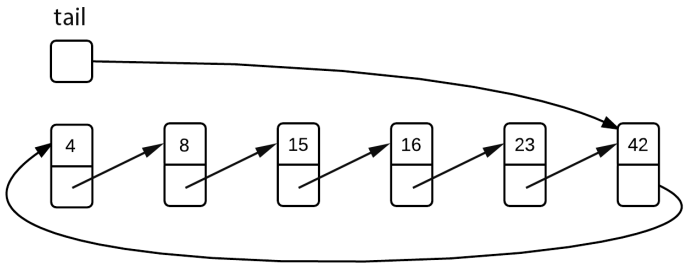
31. (2 points.) In what sense is Verizon's `X-UIDH` header a "supercookie," different from a typical cookie?

32. (2 points.) Is Verizon able to inject its `X-UIDH` header into HTTPS requests? Why or why not?

for staff use only
points off
initials

Full Circle.

Suppose that a queue for integers is implemented with a circular singly-linked list, whose tail is linked to its head, as per the example below, wherein `4` is at the front (i.e., head) of the queue and `42` is at the end (i.e., tail) of the queue. An upside of this representation is that only one global pointer (to the queue's tail), herein called `tail`, is required to maintain the structure efficiently.



Suppose that a `node` in this queue is defined per the below.

```
typedef struct node
{
    int datum;
    struct node* next;
}
node;
```

And suppose that the global pointer to the queue's tail is declared per the below.

```
node* tail = NULL;
```

33. (4 points.) Consider the buggy implementation of `dequeue`, below, that's supposed to dequeue (i.e., remove) the `datum` at the beginning (i.e., head) of the queue. Upon success, the function is supposed to store the dequeued `datum` at the address in `location` and then return `true`. Upon any error, the function is supposed to return `false`, without changing the value at the address in `location`.

    This implementation tends to work fine for a while but eventually segfaults, even when `location` is a valid pointer. Identify the source of the bug and propose via code or comments how to fix.

```
bool dequeue(int* location)
{
    if (location == NULL)
    {
        return false;
    }
    if (tail == NULL)
    {
        return false;
    }
    node* head = tail->next;
    *location = head->datum;
    tail->next = head->next;
    free(head);
    return true;
}
```

for staff use only
points off
initials

34. (6 points.) Complete the implementation of `enqueue`, below, in such a way that it adds `datum` to the end (i.e., tail) of the queue in *O*(1) time. Upon success, the function should return `true`. Upon any error, the function should return `false`, without changing the queue.

```
bool enqueue(int datum)
{
```

for staff use only
points off
initials