Quiz 1

out of 94 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.

Alisa Nguyen
Allison Buchholtz-Au
Angela Li
Armaghan Behlum
Ben Shryock
Bo Ning Han
Casey Grun
Chi Zeng
Christopher Bartholomew
Chris Mueller
Cynthia Meng
Dan Bradley
Daven Farnham
David Kaufman
Dianna Hu
Doug Lloyd
Ed Hebert
Gabriel Guimaraes
George Wu
Hannah Blumberg
Ian Nightingale
Jackson Steinkamp
Jared Pochtar
Jason Hirschhorn
Joe McCormick
Jonathan Marks
Jonathan Miller
Jordan Jozwiak
Joseph Ong
Julian Salazar
Karen Xiao
Katryna Cadle
Keenan Monks
Kendall Sherman
Kevin Mu
Kevin Schmid
Levi Roth
Lucas Freitas
Luciano Arango
Lucy Cheng
Lydia Chen
Marshall Zhang
Mike Rizzo
Patrick Schmid
R.J. Aquino
Rebecca Chen
Rhed Shi
Rob Bowden
Roger Zurawicki
Saheela Ibraheem
Sharon Zhou
Tim McLaughlin
Tomas Reimers
Travis Downs
Tyler Morrison
Varun Sriram
Wesley Chen
William Hakim
William Xiao
Zamyla Chan

for staff use only

final score out of 94

Getting Strings.

Consider three different versions of a program, below, each of which is intended to get and print a string from a user.

```
// version 1

#include <stdio.h>

int main(void)
{
    char* s;
    scanf("%s", s);
    printf("%s\n", s);
}
```

```
// version 2

#include <stdio.h>

int main(void)
{
    char s[50];
    scanf("%s", s);
    printf("%s\n", s);
}
```

```
// version 3

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    char* s = malloc(50);
    scanf("%s", s);
    printf("%s\n", s);
    free(s);
}
```

0. (2 points.) Suppose that version 1 is compiled and executed. Why might the program segfault?

1. (2 points.) Suppose that version 2 is compiled and executed. Why might the program segfault?

2. (2 points.) Suppose that version 3 is compiled and executed. Why might the program segfault?

3. (2 points.) With respect to the programs' use of memory, how do version 2 and version 3 differ?

for staff use only
—

Small Bytes.

4. (5 points.) Consider the types below.

```
typedef char* string;

typedef struct node
{
    int n;
    struct node* next;
}
node;

typedef struct
{
    char* name;
    char* house;
}
student;
```

Complete the table below by recording to the right of each type its size in bytes (not bits). Assume a 32-bit architecture like the CS50 Appliance.

type	**bytes**
`node`	
`node*`	
`string`	
`struct node*`	
`student`	

for staff use only
—

HTTP/1.1.

For each of the HTTP status codes below, explain, in no more than three sentences, why a server might respond with that code.

5. (2 points.) `200 OK`

6. (2 points.) `302 Found`

7. (2 points.) `403 Forbidden`

8. (2 points.) `404 Not Found`

9. (0 points.) `418 I'm a teapot`

10. (2 points.) `500 Internal Server Error`

for staff use only
—

Double Trouble.

Consider the program below, wherein `list` is an unsorted doubly-linked list of integers.

```
#include <stdbool.h>
#include <stdlib.h>

typedef struct node
{
    int n;
    struct node* prev;
    struct node* next;
}
node;

node* list = NULL;

void insert(int n);
void remove(int n);

int main(void)
{
    insert(50);
    insert(34);
    insert(59);
    insert(20);
    insert(106);
    insert(36);
    insert(52);
    remove(50);
}
```

11. (4 points.) Complete the implementation of `insert` below in such a way that it inserts `n` into `list` in constant time (even if already present).

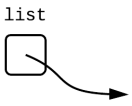
```
void insert(int n)
{
```

for staff use only
—

12. (6 points.) Complete the implementation of `remove` below in such a way that it removes `n` from `list` in linear time. You may assume that `n`, if present in `list`, will be present no more than once.

```
void remove(int n)
{
```

13. (2 points.) Suppose that execution of the program is paused after all statements in `main` have executed, per the prescribed implementations of `insert` and `remove`, but before `main` returns (i.e., exits). Complete the drawing below in such a way that it depicts the unsorted doubly-linked list that exists in memory at that moment in time.



for staff use only
—

World Wide What.

14. (6 points.) Suppose that a user types `https://www.facebook.com/` into his or her browser's address bar and then hits Enter (after clearing any cookies). Explain, in a short paragraph, the process by which Facebook's home page is retrieved, using each of the six 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 · IP address · port · router

Quadded.

15. (2 points.) Consider the PHP array below.

```
$quad = ["Cabot House", "Currier House", "Pforzheimer House"];
```

Complete the code fragment below in such a way that it outputs an unordered (i.e., bulleted) list of houses by iterating over `$quad`. Assume that `$quad` is in scope. Take care to close the `ul` tag that we've opened for you, but no need to add elements like `html`, `head`, or `body`. Recall that `ul` elements have `li` (i.e., list item) elements as children.

```
<ul>
    <?php
```

for staff use only
—

Tradeoffs. (2 points each.)

For each of the design decisions below, explain one upside and one downside, each in no more than two sentences.

16. Validating users' input client-side (as with JavaScript) instead of server-side (as with PHP).

    upside:

    downside:

17. Using a linked list instead of an array to maintain a sorted list of integers.

    upside:

    downside:

18. Using PHP instead of C to write a command-line program.

    upside:

    downside:

19. Using a SQL database instead of CSV file to store data.

    upside:

    downside:

20. Using a trie instead of a hash table with separate chaining to store a dictionary of words.

    upside:

    downside:

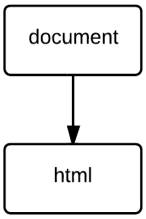
for staff use only
—

DOM, DOM DOM DOM.

21. (4 points.) Consider the HTML below.

```
<!DOCTYPE html>

<html>
    <head>
        <title>quiz</title>
    </head>
    <body>
        <div>top</div>
        <div>middle</div>
        <div>bottom</div>
    </body>
</html>
```

Complete the DOM below in such a way that it represents the HTML above, using any shape(s) for nodes.



for staff use only
—

Attack!

Recall the program below.

```
#include <string.h>

void foo(char* bar)
{
    char c[12];
    memcpy(c, bar, strlen(bar));
}

int main(int argc, char* argv[])
{
    foo(argv[1]);
}
```

22. (2 points.) How might an adversary "trick" this program into executing arbitrary code?

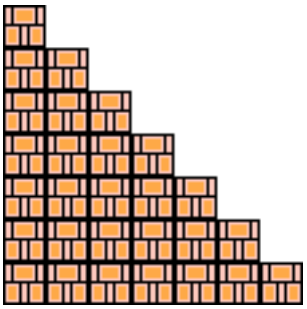
23. (3 points.) Complete the re-implementation of `foo` below in such a way that the program is no longer vulnerable to that attack. Any approach that prevents the attack is acceptable.

```
void foo(char* bar)
{
```

for staff use only
—

Itsa Mario again?

24. (4 points.) Toward the end of World 2-3 in Nintendo's Super Mario Brothers 3, Mario must descend a "half-pyramid" of blocks (unless he flies over it). Below is a screenshot.



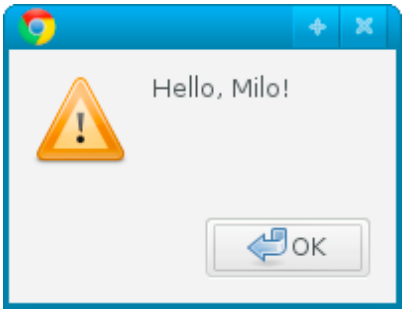
Complete the implementation of the command-line PHP program below in such a way that it recreates this particular half-pyramid using hashes (`#`) for blocks. No need for user input; you may hard-code the half-pyramid's height (`7`) into your program.

```
#!/bin/env php
<?php
```

for staff use only
—

woof am back.

25. (4 points.) Complete the implementation of the web page below in such a way that if someone visits the page, inputs his or her name, and then submits the form, he or she is greeted via a call to `alert` with a window like that at right. If a user fails to input a name before submitting the form, no alert should appear. You are welcome, but not required, to use jQuery functions.




```
<!DOCTYPE html>

<html>
    <head>
        <script src="http://code.jquery.com/jquery-latest.min.js"></script>
        <script>









        </script>
        <title>quiz</title>
    </head>
    <body>
        <form id="inputs">
            <input id="name" type="text"/>
            <input id="button" type="submit" value="Greet"/>
        </form>
    </body>
</html>
```

for staff use only
—

MyOhMySQL.

Suppose that a bank relies on a SQL table called `accounts` to store customers' accounts, wherein `number` (the table's `PRIMARY` key) represents a user's 12-digit account number, and `balance` represents how much money the customer has in his or her account, per the (incomplete) schema below. As per the default value of `100.00` for `balance`, the bank gives customers $100 anytime they open an account.

Name	Type	Null	Default
number		No	*None*
balance		No	100.00

26. (2 points.) Complete the schema for `accounts` above by specifying, next to `number` and `balance` on the schema itself, an appropriate SQL type for each.

27. (2 points.) Suppose that the bank imposes a $20 "monthly maintenance fee" on all accounts. With what SQL query could the bank deduct $20 from every account (even if it results in some negative balances)?

28. (2 points.) With what SQL query could the bank retrieve the account numbers of its richest customers, those with balances greater than $1,000?

29. (2 points.) With what SQL query could the bank close (i.e., delete) every account that has a balance of $0?

for staff use only
—

Queue Guide.

Suppose that a queue for (non-negative) integers is defined per the below, wherein `CAPACITY` (a global constant) is the maximum number of integers that can be in the queue, `size` is the number of integers currently in the queue, and `front` is the index of the front of (i.e., first integer in) the queue.

```
typedef struct
{
    int numbers[CAPACITY];
    int front;
    int size;
}
queue;
```

Assume that a `queue` has been declared globally with

```
queue q;
```

and that `q` has been initialized for you (e.g., in `main`) per the below.

```
q.front = 0;
q.size = 0;
```

30. (4 points.) Complete the implementation of `enqueue` below in such a way that the function adds `n` to (the end of) `q` and then returns `true`. If `q` is full or `n` is negative, the function should instead return `false`. Assume that `bool` as well as `true` and `false` have been declared for you (as via `cs50.h` or `stdbool.h`).

```
bool enqueue(int n)
{
```

31. (4 points.) Complete the implementation of `dequeue` below in such a way that the function dequeues (i.e., removes and returns) the `int` at the front of `q`. (To remove the `int`, it suffices to "forget" it; you needn't overwrite its bits.) If `q` is empty, the function should instead return `-1`.

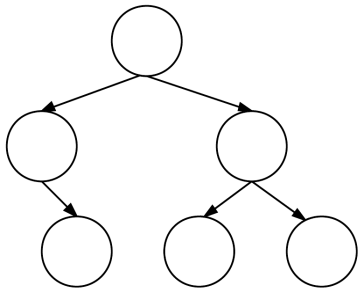
```
int dequeue(void)
{
```

for staff use only
—

Pssst, BST.

Recall that a binary search tree (BST) is a binary tree (each of whose nodes thus has 0, 1, or 2 children), the value of whose root is: greater than that of its left child, if any, and any descendants thereof; and less than that of its right child, if any, and any descendants thereof. Moreover, each child is the root of a BST.

32. (2 points.) Consider the binary tree below whose nodes do not yet have values.



Insert the six integers below into that tree, as by writing one integer in each circle, in such a way that the binary tree is a BST.

34 · 59 · 20 · 106 · 36 · 52

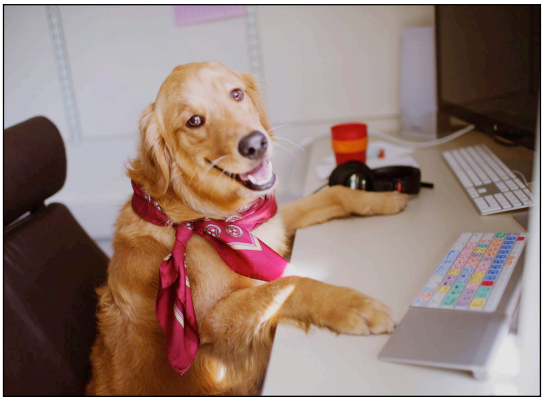
33. (2 points.) Complete the definition of `node` below in such a way that it represents a node in a BST, the value of which is of type `int`. You may name its fields however you'd like.

```
typedef struct node
{









}
node;
```

for staff use only
—

34. (4 points.) Complete the implementation of `traverse` below in such a way that, given the root of a BST, the function performs an "in-order traversal," printing each of the tree's integers with `printf` from smallest to largest, one per line. Assume that `node` is defined per your own implementation one problem ago. Do not assume that `root` will be non-`NULL`.

```
void traverse(node* root)
{
```



for staff use only
—

Scrap Paper.

Nothing on this page will be examined by the staff unless otherwise directed.