Quiz 1

Answer Key

Answers other than the below may be possible.

Multiple Choice.

0. c
1. d
2. a
3. b

True or False.

4. F
5. F
6. F

DOM, DOM DOM DOM.

7.
```
<!DOCTYPE html>

<html>
    <head>
        <title>CS50</title>
    </head>
    <body>
        <h1>CS50</h1>
        <h3>Under Construction</h3>
        <h6>Copyright 2012</h6>
    </body>
</html>
```

`GetInt` 1.0.

8. It reads an `int` from standard input and stores it in `n`.

9. So that `scanf` can alter the value of `n` and not a copy thereof.

`GetString`.

10. Even though `s` is a pointer, it's never initialized with the address of a chunk of memory, and so `scanf` ends up writing characters from standard input to some garbage value (i.e., memory that doesn't belong to `GetString`), the result of which is often a segfault.

`GetInt` 2.0.

11. `0`

12. The `%c` placeholder instructs `sscanf` to try reading a `char` (into `c`) after reading an `int` (possibly after some whitespace). If the user has indeed inputted more than just an `int`, `c` will be filled with the first such `char` (that isn't a digit or whitespace), and so `sscanf` will return `2`, thereby indicating that the user did not, in fact, input only an `int`.

13. To reserve `1` or `-1` as a "sentinel" value indicating failure would mean that users couldn't actually type `1` or `-1` as input. Inasmuch as users are more likely to want to type `1` or `-1` than `2147483647`, reserving the latter ensures that `GetInt` is usable for the more common cases.

Register here.

14.
```
<html>
    <head>
        <title>Register</title>
    </head>
    <body>
        <?php

            if (empty($_POST["email"]) || empty($_POST["password"]) || empty($_POST["confirmation"]))
                echo "FAILURE";
            else if ($_POST["password"] != $_POST["confirmation"])
                echo "FAILURE";
            else
                echo "SUCCESS";

        ?>
    </body>
</html>
```

15.
```
<!DOCTYPE html>

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

            // onready
            $(document).ready(function() {

                // onsubmit
                $('#registration').submit(function() {

                    if ($('#email').val() == '' || $('#password').val() == '' || $('#confirmation').val() == '')
                        return false;
                    else if ($('#password').val() != $('#confirmation').val())
                        return false;
                    else
                        return true;

                });
            });

        </script>
        <title>Register</title>
    </head>
    <body>
        <form action="register.php" method="post" id="registration">
            Email: <input id="email" name="email" type="text"/>
            <br/>
            Password: <input id="password" name="password" type="password"/>
            <br/>
            Password (again): <input id="confirmation" name="confirmation" type="password"/>
            <br/><br/>
            <input type="submit" value="Register"/>
        </form>
    </body>
</html>
```

Under attack.

16. A buffer overflow is the act of writing data beyond the boundaries of some block of allocated memory (e.g., an array), whether intentionally or unintentionally.

17. `C`

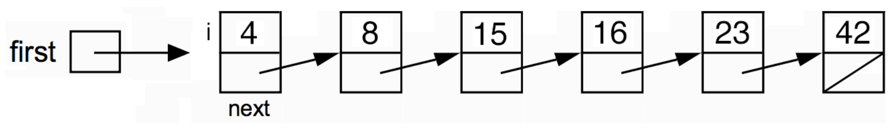
18. If an adversary overflows some buffer by providing more bytes than were anticipated, some of which collectively represent executable code, and the adversary successfully overwrites some "return address" on the program's stack with the address of that executable code, then a program might be tricked into executing it by "returning" to it from some function.

Pointer fun with singly linked lists.

19. *O*(*n*)

20. Ω(1)

21.



Pointer fun with doubly linked lists.

22.
```
typedef struct node
{
    struct node* prev;
    unsigned int i;
    struct node* next;
}
node;
```

23.
```
void insert(unsigned int i)
{

    // try to instantiate node for number
    node* n = malloc(sizeof(node));
    if (n == NULL)
        return;

    // initialize node
    n->prev = NULL;
    n->i = i;
    n->next = NULL;

    // check for empty list
    if (first == NULL)
        first = n;

    // else check if number belongs at list's head
    else if (n->i < first->i)
    {
        n->next = first;
        first->prev = n;
        first = n;
    }

    // else try to insert number in middle or tail
    else
    {
        node* ptr = first;
        while (true)
        {
            // avoid duplicates
            if (ptr->i == n->i)
            {
                free(n);
                return;
            }

            // check for insertion at tail
            else if (ptr->next == NULL)
            {
                ptr->next = n;
                n->prev = ptr;
                return;
            }

            // check for insertion in middle
            else if (ptr->next->i > n->i)
            {
                n->next = ptr->next;
                ptr->next->prev = n;
                n->prev = ptr;
                ptr->next = n;
                return;
            }

            // advance pointer to next node
            ptr = ptr->next;
        }

    }
}
```

Such a grind.

24. The programmer has likely written to a 4-byte location in memory that does not belong to his or her program, as by indexing beyond the boundary of an array of `int`s (on a 32-bit machine like the CS50 Appliance).

25. The programmer has likely allocated 40 bytes of memory (as by allocating 10 `int`s on a 32-bit machine like the CS50 Appliance) but has failed to free them.

All your base.

26.

Binary	**Decimal**	**Hexadecimal**
00000000	0	0x00
00100000	32	0x20
00110010	50	0x32
11011111	223	0xdf

Bold Claims.

27. Incorrect. HTML is a markup language that allows you to describe the structure or semantics of a web page. It doesn't allow you to express control flow or logic.

28. Correct. For small files (or files with a fairly uniform distribution of most ASCII characters), the overhead of storing the tree (or equivalent) might very well outweigh the savings in bits.

29. Incorrect. GIF is a lossless format that decreases an image's size by using fewer bits to represent the same amount of information, as by recognizing when adjacent pixels are the same color and storing a summary rather than storing the color of each pixel individually.

30. Incorrect. Even if a program's source code is free of backdoors, it might have been compiled with a compiler that inserts a backdoor.

BSTs.

31.
```
bool find(node* root, int i)
{
    if (root == NULL)
        return false;
    else if (i < root->i)
        return find(root->left, i);
    else if (i > root->i)
        return find(root->right, i);
    else
        return true;
}
```

32.
```
bool find(node* root, int i)
{
    node* ptr = root;
    while (ptr != NULL)
    {
        if (i < ptr->i)
            ptr = ptr->left;
        else if (i > ptr->i)
            ptr = ptr->right;
        else
            return true;
    }
    return false;
}
```

Nom nom nom.

33. `http://www.cs50.net/quizzes/`

34. `PHPSESSID` is a cookie whose value is string that uniquely identifies a user's browser. After receipt of that cookie, a browser, by nature of HTTP, will include that cookie's value in all subsequent requests to the website that set it. That value maps, server-side, to a file (or database row) that contains the contents of `$_SESSION`, a PHP superglobal in which a website can store key-value pairs.

How odd.

35.
```
bool odd(unsigned int n)
{
    if (n & 1)
        return true;
    else
        return false;
}
```

Let's talk about compilers.

36. During pre-processing... directives like `#include` are processed, which, in this case, means that

    ```
    #include <stdio.h>
    ```

    is effectively replaced with the contents of `stdio.h`, thereby providing a prototype for `printf`.

    During compiling... the pre-processed C code is translated into assembly instructions, possibly with optimizations applied.

    During assembling... the assembly instructions are translated into machine code and stored in an object (`.o`) file.

    During linking... the object code for main is combined with (i.e., linked against) that for `printf`, the result of which is an executable file.

Rapid Fire.

37. The queries encapsulated by the transaction are either all executed or not at all.

38. An associative array is a data structure that allows programmers to associate values with keys, the latter of which can be numbers or even strings.

39. External stylesheets allow styles to be centralized so that changes can be made easily in one location. External stylesheets can also decrease pages' size by factoring out common properties that might otherwise be written in multiple locations.

40. MVC (model-view-controller) is a "design pattern" that separates "business logic" (C) from the representation of data (M) from the display of that data (V). Put another way, a controller controls an application's behavior, taking input from users and responding accordingly. And a view is the output that's ultimately presented to a user by a controller.

CS50 Associates.

41.
```
<table>
    <tr>
        <td>Name</td>
        <td>House</td>
    </tr>
    <?php

        foreach ($tfs as $tf)
        {
            print("<tr>");
            print("<td>{$tf["name"]}</td>");
            print("<td>{$tf["house"]}</td>");
            print("</tr>");
        }

    ?>
</table>
```

Sorting students.

42. `http://hogwarts.edu/lottery.php?name=Harry&house=Gryffindor`

Storing students.

43. A data type of `INT` (possibly `UNSIGNED`) would work well. Not only do houses' identifiers already appear to be numeric, an `INT` would allow us to `JOIN` the two tables efficiently. Alternatively, because Harvard has relatively few houses, even a `TINYINT` would suffice.)

44. A data type of `VARCHAR` (whose maximal size is at least `10`) would work well, since it would allow houses' names to be of disparate lengths without wasting (much) space. Alternatively, because the existing houses' names are so close in length, even `CHAR` would work well, since very few bytes would be wasted.

45. A data type of `INT` (possibly `UNSIGNED`) would work well, as would even `TINYINT`, so long as it's identical to that used for `id` in `houses`.

46. `UPDATE students SET house = NULL WHERE id = 4`

47. `INSERT INTO students (name, house) VALUES('Luna', 3)`

48. `SELECT * FROM students JOIN houses ON students.house = houses.id`