Quiz 1

Answer Key

Answers other than the below may be possible.

True or False.

0. T or F
1. T
2. T
3. T

DOM DOM DOM.

4.
```
<!DOCTYPE html>

<html>
  <head>
    <title>Houses</title>
  </head>
  <body>
    <h1>Houses</h1>
    <ul>
      <li>Mather</li>
      <li>other</li>
    </ul>
  </body>
</html>
```

Bah.

5. Any website that transmits session cookies in the clear is vulnerable, not just Facebook.

6. Firesheep doesn't try to guess session cookies, it intercepts actual ones by listening on a local network for unencrypted ones from known sites.

7. Facebook does not use SSL for other pages, though, and so your once-encrypted session cookie is subsequently transmitted in the clear.

8. Even though the connection between you and Harvard's VPN server is encrypted, that between Harvard's VPN server and Facebook itself is not. And so someone on the Internet between Harvard and Facebook could still hijack your sessions.

Frosh IMs.

9.
```
http://froshims.net/register.php?name=David&gender=M&dorm=matthews
```

10.
```
function validate()
{
    if (document.getElementById("name").value == "David")
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Shuttletime.

11.
```
bool
taken(int seat)
{
    if (seat < 0 || seat >= SEATS)
        return false;
    return seats[seat];
}
```

12.
```
bool
take(int seat)
{
    if (seat < 0 || seat >= SEATS || seats[seat] == true)
        return false;
    seats[seat] = true;
    return true;
}
```

13. `5`

14. 5

15.
```
for (int i = 0; i < BYTES; i++)
    seats[i] = 0x00;
```

16.
```
bool
taken(int seat)
{
    if (seat < 0 || seat >= SEATS)
        return false;
    int byte = seat / 8;
    int mask = 1 << seat % 8;
    return seats[byte] & mask;
}
```

17.
```
bool
take(int seat)
{
    if (seat < 0 || seat >= SEATS)
        return false;
    int byte = seat / 8;
    int mask = 1 << seat % 8;
    if (seats[byte] & mask)
        return false;
    seats[byte] = seats[byte] | mask;
    return true;
}
```

Pointer Fun, still without Binky.

18. The function prints the given string, one character per line.

Numeric Self Defense.

19. Plausible. If the amount of RAM in laptops has doubled every 18 months (*i.e.*, 1.5 years) since 1996, then it's doubled $14/1.5 \approx 9$ times, since 1996 was 14 years ago. David had 4MB then, so he should have $4\text{MB} \times 2^9 = 2048\text{MB}$ (*i.e.*, 2GB) now, which is indeed plausible.

20. Much too high. Even if only 1 student had enrolled in CS50 in 1989, 21 years have passed, which would imply a current enrollment on the order of $1 \times 2^{21} = 2{,}097{,}152$.

Design Decisions.

21. JavaScript should be used when you want to execute code client-side, perhaps to handle user input, manipulate a web page's DOM, or induce subsequent HTTP requests. PHP should be used when you want to execute code server-side, perhaps to handle a form's submission, generate HTML, or write to a database.

22. You should pass an argument by reference when you want the callee to be able to change it or when the argument is a multi-byte `struct` that you'd like to avoid copying (which takes time and space). You should pass an argument by value when you don't want the callee to be able to change it or when the time and space involved in copying it is negligible.

23. You should use `gdb` when you want to debug code, as by setting breakpoints, stepping through statements, and examining memory. You should use `valgrind` when you want to chase down memory leaks and invalid pointer dereferences.

24. The size of a `long` varies by architecture, so you should use an `int64_t` when you want to ensure that a variable is a 64-bit signed value.

25. You should use POST when a form needs to submit more data than would reasonably fit in a URL or when a form's data warrants privacy (as do passwords). You should use GET when you want a web page and its state to be bookmarkable or emailable.

Bases Covered.

26.

Binary	**Decimal**	**Hexadecimal**
00000000	0	0x00
00000010	2	0x02
00001010	10	0x0A
00010000	16	0x10

Structures.

27. `1233`

28. `1337`

More Structures.

29. Even though searching a hash table and searching a linked list might be asymptotically equivalent, the fact remains that, in the real world, the former might very well take 1/26 as much time as the latter, a difference that humans might certainly notice and appreciate.

30.
```
void
print_r(node *tree)
{
    if (tree == NULL)
        return;
    print_r(tree->left);
    if (tree->left)
        printf(",");
    printf("%d", tree->n);
    if (tree->right)
        printf(",");
    print_r(tree->right);
}
```

Axe to Valgrind.

31. Cansu has likely written to a 4-byte location in memory that does not belong to her program, as by indexing beyond the boundary of an array of `int`s.

32. Yuhki has likely allocated 40 bytes of memory (as by allocating 10 `int`s) but failed to free them.

Quickies.

33. Steganography is the science of hiding information, as by manipulating the pixels in an image in such a way that their colors represents ASCII values.

34. With external CSS files can you factor out properties that might be common to multiple HTML elements.

35. Two-factor authentication involves challenging users to present two forms of identification in order to proceed, generally something they know (*e.g.*, a password) and something they have (*e.g.*, a keyfob).

36. `XMLHttpRequest` objects enable Ajax, a technique whereby a web page can make HTTP requests to a server programmatically, often in response to user input, in order to integrate new content into its DOM.

37. `NULL` is a special pointer that points to no object, whereas `'\0'` is a `char` that generally demarks the end of a string.

38. An HTTP cookie is a key-value pair planted by a web server on a user's computer, either in RAM or on disk.

39. The Birthday Problem reveals just how likely collisions are, even when relatively few inputs are distributed uniformly over a finite number of buckets.

40. An event handler is a function (or method) that's called in response to some event, such as the click of a mouse.

Hello, Katie.

41. Version 1 is incorrect, as `printf` expects a pointer to a char as its second argument, as implied by the format code in its first argument. But `*s` denotes a `char`, which is really just a number, and so `*s` will be incorrectly interpreted as an address, and so the function call may very well segfault or, at least, print garbage values.

    Version 2 is correct.

    Version 3 is incorrect. As in Version 1, `printf` expects a pointer to a char as its second argument, but `&s` denotes the address of the address of a `char`, and so the function call may very well segfault or, at least, print garbage values.

My oh my, SQL.

42. `UPDATE clients SET cash = cash – 20 WHERE cash < 5000`

43. `DELETE FROM clients WHERE username='dshen'`

44. The hacker's input will result in construction of

```
SELECT id FROM clients WHERE username='$username' AND password='' OR '1' = '1'
```

which is equivalent to

```
SELECT id FROM clients
```

since `'1'` is indeed equal to `'1'`. And so the query will return all clients, in which case

```
mysql_num_rows($result)
```

will not return `0` (assuming the broker has at least one client), at which point the first such client's `id` will be stored as the value `$_SESSION["id"]`. The result, presumably, is that the hacker will effectively be logged in as that user, even without having known his or her password.

To fix, it suffices to escape all user input, as with

```
$username = mysql_real_escape_string($_POST["username"]);
$password = mysql_real_escape_string($_POST["password"]);
```

since `mysql_real_escape_string` will prefix single quotes with backslashes.