This is CS50.
quiz 1 details

wed nov 17, 1pm
see handout for locations
covers weeks 0 through 10, emphasis on 7 onward
closed book
bring a 8.5” × 11”, 2-sided cheat sheet
75 minutes
15% of final grade
resources

old quizzes + solutions
lecture slides
lecture videos + transcripts
source code
scribe notes
section videos
pset specs
office hours
quiz 0 review
topics
review
Part 0

Josh Bolduc
bitwise operators
**bitwise AND**

I iff both bits are 1

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&amp;</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>&amp;</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>&amp;</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>&amp;</td>
<td>1</td>
</tr>
</tbody>
</table>

```
01011110  
& 11001000  
___________
01001000  
```
**bitwise OR**

iff one or both bits are 1

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
01011110 \\
\text{|} 11001000 \\
\text{|} 11011110
\end{array}
\]
**bitwise XOR**

iff exactly one bit is 1

<table>
<thead>
<tr>
<th></th>
<th>^</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

01011110

^ 11001000

11001000

10010110
ones complement

1 iff 0

\[
\begin{array}{|c|c|}
\hline
\sim 0 & 1 \\
\hline
\sim 1 & 0 \\
\hline
\end{array}
\]

\[\sim 01011110\]

\[10100001\]
left shift
multiply by $2^n$

<table>
<thead>
<tr>
<th>$01001101$ $\ll$ $n$</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$01001101$ $\ll$ $0$</td>
<td>$01001101$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $1$</td>
<td>$10011010$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $2$</td>
<td>$00110100$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $3$</td>
<td>$01101000$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $4$</td>
<td>$11010000$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $5$</td>
<td>$10100000$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $6$</td>
<td>$01000000$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $7$</td>
<td>$10000000$</td>
</tr>
<tr>
<td>$01001101$ $\ll$ $8$</td>
<td>$00000000$</td>
</tr>
</tbody>
</table>
right shift
divide by $2^n$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01001101 &gt;&gt; 0</td>
<td>01001101</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 1</td>
<td>00100110</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 2</td>
<td>00010011</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 3</td>
<td>00001001</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 4</td>
<td>00000100</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 5</td>
<td>00000010</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 6</td>
<td>00000001</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 7</td>
<td>00000000</td>
</tr>
<tr>
<td>01001101 &gt;&gt; 8</td>
<td>00000000</td>
</tr>
</tbody>
</table>
## base $n$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>base 2</td>
</tr>
<tr>
<td>octal</td>
<td>base 8</td>
</tr>
<tr>
<td>decimal</td>
<td>base 10</td>
</tr>
<tr>
<td>hexadecimal</td>
<td>base 16</td>
</tr>
</tbody>
</table>
hexadecimal

\[ \text{0x } \quad \_\_\_ \_ \_ \_ \_ \_ \_ \_ \]

\[ 16^7 \quad 16^6 \quad 16^5 \quad 16^4 \quad 16^3 \quad 16^2 \quad 16^1 \quad 16^0 \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

1 hex digit == 4 bits
2 hex digits == 8 bits == 1 byte
$\text{0x25ba}$
(base 16)

\[
\begin{align*}
2 \times 16^3 & + 5 \times 16^2 \\
+ 11 \times 16^1 & + 10 \times 16^0 \\
\end{align*}
\]

9658
(base 10)
42
(base 10)

2 \times 16^1 \rightarrow 2
10 \times 16^0 \rightarrow a

0x2a
(base 16)
endianness

(int) 0x1a2b3c4d

little-endian
(lowest order bytes first)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
4d & 3c & 2b & 1a \\
\end{array}
\]

big-endian
(highest order bytes first)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
1a & 2b & 3c & 4d \\
\end{array}
\]
stacks
typedef struct {
    int numbers[CAPACITY];
    int size;
    int top;
} stack;
queues
typedef struct {
  int head;
  int numbers[CAPACITY];
  int size;
} queue;
Part I

Tian Feng
Hash Tables
Hash Tables

- Almost constant time access
- Maps value of key object to some storage index $i$ in the hash table array
- In pset 6, this mapped a string key to an int $i$
Hash Functions

• Good hash functions:
  – Return the same code for identical strings every time
  – Provides a uniform distribution of hash values (if desired)

• Potential problems:
  – Collisions
  – Clustering
Separate Chaining

```
int hash(char *s)
{ return s[0] - 'a'; }
```

Cat: ‘c’ – ‘a’ = 2
Art: ‘a’ – ‘a’ = 0
Car: ‘c’ – ‘a’ = 2
Con: ‘c’ – ‘a’ = 2
Giant: ‘g’ – ‘a’ = 6
[Linear] Probing

```
int hash(char *s)
{return s[0] - 'a';}
```

Cat: ‘c’ – ‘a’ = 2
Art: ‘a’ – ‘a’ = 0
Car: ‘c’ – ‘a’ = 2
  but wait, 2 is taken
  continue to 3
Con: ‘c’ – ‘a’ = 2
  but wait, 2 is taken
  but wait, 3 is taken
  continue to 4
Giant: ‘g’ – ‘a’ = 6
Trees and Tries
typedef struct node {
    int value;
    struct node *child1;
    struct node *child2;
    struct node *child3;
} node;

Trees

typedef struct node {
    int value;
    struct node *child1;
    struct node *child2;
    struct node *child3;
} node;
Tree Terminology

- Root
- Parent node
- Child node
- Leaf node
- Internal node
- Height
- Depth
Binary Search Trees

typedef struct node {
    int value;
    struct node *left;
    struct node *right;
} node;
typedef struct node {
    bool isword;
    struct node *array[27];
} node;

Figure from Lewis and Denenberg’s
Data Structures & Their Algorithms.
Part 2
Rose Cao
The Acronyms

- HTTP
- HTML
- CSS
HyperText Transfer Protocol

- Rules to get linked resources
- Client Requests, Server Responds
HyperText Transfer Protocol

• Request ->
  – Domain Name System lookup
    • Maps URL to IP address
  – Routed to IP address
  – GET / HTTP/1.1

• IRL example:
  GET /~hcat/pset7/login.php HTTP/1.1
  POST /~hcat/pset7/login2.php HTTP/1.1
  GET /~hcat/pset7/index.php HTTP/1.1
HyperText Transfer Protocol

- Response
  - Displaying instructions (HTML)
  - Graphics
  - Markup

- Browser renders the HTML
HyperText Markup Language

• Interpreted language
  – By browser

• Two main sections:
  <head>
    • title, script declarations
  </head>

  <body>
    • page contents
  </body>
HyperText Markup Language

• Tags
  – Elements
  – Attributes & Attribute Values

Ex: `<img src = "ohai.jpg" />`

• A few other useful tag elements:
  `<div>`, `<form>`, `<table>`
• Allows for (easier) HTML formatting
• Can be in the page source code or in a separate .css file
• Separates content from presentation
  – Specific formatting for different page elements
  – Selectors
Cascading Style Sheets

• Basic Selector:

```
body {
    background: black;
    color: white;
}
```
Cascading Style Sheets

• ID Selectors:
  – For a single, unique id

    #ans
    {
      font-weight: bold
      color: blue;
    }

    <div id = “ans”> 42! </div>
Cascading Style Sheets

• Class Selector:
  – For a group of elements

```css
.center
{
  text-align: center;
}
```

```html
<table class = "center"> ... </table>
<div class = "center"> ... </div>
```
Cascading Style Sheets

• Including in document:

```html
<head>
  <style type = “text/css”>
    <!--
    body
    {
      ...
    -->
    </style>
  ...
</head>
```

• Including in separate file:

```html
<head>
  <link href = “styles.css” rel = “stylesheet”
       type = “text/css”>
  ...
</head>
```
PHP and SQL
PHP Hypertext Preprocessor

• The name is recursive!! How awesome is that.

• Allows us to create a backend for our website. i.e. the logic

• Loosely Typed
Defining a Variable in PHP

• No need to specify a type

```php
$foo = 3;
$bar = 6;
$baz = $foo + $bar;
$string1 = “Apple”;
$float1 = 0.205;
```
String Concatenation

• The dot operator!

```php
$string0 = "Apple" . " is " . "awesome";

$string1 = "Apple";
$string2 = "iPad";

$string3 = $string1 . " " . $string2;
```
GET

• Is a retrieval method, for information that is not too sensitive. Displayed in URL.

HTML:

<form action="buy2.php" method="get">
    <input name="symbol" type="text">
</form>

PHP:

$symbol = mysql_real_escape_string($_GET["symbol"]);
• Marginally more private (info not in url)

HTML:

<form action="sell2.php" method="post">
  <input name="shares" type="text">
</form>

PHP:

$shares = mysql_real_escape_string($_POST["shares"]);
$_SESSION

• Allows us to retrieve session information about the current HTTP session.

$uid = $_SESSION[“uid”];
echo

With echo you can print to the screen.

```php
$foo = "hello";
$bar = "hi";

echo $foo; //prints hello
echo '$foo'; //prints $foo
echo $foo,$bar; //prints hellohi
```
Arrays in PHP

Arrays in PHP have both a key and a value:
- Keys can be only an integer or a string.

```php
$myArray = array("key1" => 1, 4 => "hello");

$value1 = $myArray["key1"];  // value1 = 1
$value2 = $myArray[4];       // value2 = hello
```

Having an array with different types of keys is not ideal!
SQL

• Structured Query Language

• Think of a SQL database as a collection of tables that you can manipulate: select, insert, delete, update

• Using the * in your query means that you want every field.
Our little database

The Fields:

uid: int (Primary Key)
name: varchar
age: int
password: varchar
Our Devoted Users

<table>
<thead>
<tr>
<th>uid</th>
<th>name</th>
<th>age</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>steve</td>
<td>55</td>
<td>osx</td>
</tr>
<tr>
<td>2</td>
<td>bill</td>
<td>55</td>
<td>dos</td>
</tr>
</tbody>
</table>
SELECT

$query1 = "SELECT age, password FROM users WHERE uid = 1";

$sql1 = mysql_query($query1);

//using arrays
$row = mysql_fetch_array($sql1);
$age = $row["age"];

$age += 1;  //increments steve’s age by 1 year
$age = 60;
$password = 'password';
$name = 'larry';

$query2 = "INSERT INTO users (name, age, password) VALUES ('$name', $age, '$password');";

$sql2 = mysql_query($query2);
Let’s delete bill from our users!

$query3 = "DELETE FROM users WHERE uid = 2";

$sql3 = mysql_query($query3);
UPDATE

Now let’s say steve’s is logged in and changes his age to 56

$query4 = mysql_query("UPDATE users SET age = age + 1 WHERE uid = 
{$_SESSION["uid"]}");
Part 4

Matthew Chartier
Up Next…

• Javascript
• Ajax
• Language Review
JavaScript

Language Properties

• Interpreted
• Loosely Typed
• Client-Side Execution
JavaScript

Variables

- Global by default - `x`
- Local when specified – `var x`
- Need not specify type!
JavaScript

Arrays

• May contain any number of elements, varying types of values.

• \( x = [19, \text{true}, \text{null}, "\text{llama}", 42.0]; \) is okay!

• Overreaching bounds of array yields 'undefined' (No SegFaults!)
JavaScript

Data Types

- Number
- Boolean
- String
- Object
- Function
JavaScript

Data Types

typeof x – returns type of x

== : compare values, disregard type

“123” == 123 ✓

=== : compare values and type

“123” === 123 X
JavaScript

Event Handlers
JavaScript

Event Handlers

• Functions may be called in response to 'events'
• Interpreter wait for events to occur, then calls the handler functions which are assigned to them
• Events may be included in HTML tags
• Examples: onclick, onload, onsubmit...
Ajax

- Asynchronous Javascript and XML
- Allows page to dynamically fetch data from other pages
- Response information may be dynamically included in your page without refreshing
Ajax

Creation of XMLHttpRequest

• XMLHttpRequest - most browsers
• ActiveXObject – Internet Explorer

• Error handling for those less compatible
Ajax

Steps for Creating XHR Object

1. Construct URL
2. Set up event handler
3. Open the request
4. Send the request
Ajax

onreadystatechange Handler

1. Check if readyState is 4
2. Check if status is 200
3. Use responseText or responseXML
Example Ajax Call

// an XMLHttpRequest
var xhr = null;
function get_info()
{
    // instantiate XMLHttpRequest object
    try
        xhr = new XMLHttpRequest();
    catch (e)
        xhr = new ActiveXObject("Microsoft.XMLHTTP");

    // handle old browsers
    if (xhr == null)
    {
        alert("Ajax unsupported!");
        return;
    }

    // construct the URL to send to based on form contents
    var url = "ajaxhelper.php";

    // send out the xhr
    xhr.onreadystatechange = handler;
    xhr.open("GET", url, true);
    xhr.send(null);
}
Example Ajax Call

```javascript
function handler()
{
    // only deal with requests which have been loaded
    if (xhr.readyState == 4)
    {
        // check if request was successful
        if (xhr.status == 200)
        {
            // throw the response text on the page
            document.getElementById("output").innerHTML = xhr.responseText;
        }
        else
        {
            alert("XHR Fail.");
        }
    }
}
```
Language Review

C
• Compiled
• Strictly Typed

PHP
• Interpreted
• Loosely Typed
• Server-side execution
Language Review

JavaScript

- Interpreted
- Loosely Typed
- Client-side execution
Questions

?
Good Luck!