Disclaimer

- This is what I have seen students in the labs struggling with and what I think is important. This doesn’t necessarily reflect what will be on the final exam.
Overview

• Labs
• Decimal <-> Binary <-> Hexadecimal
• HTML/CSS
• JavaScript
• Q&A
Lab 0

• Got used to the basic lab environment (Piazza, ugrad email etc.)

• Learned Unix (refer to the Unix exercises at http://www.ugrad.cs.ubc.ca/~cs101/2014W1/unix.shtml)
Lab 1

- HTML/CSS (more in the HTML/CSS section)
- Learned about tags
- Put files onto the server (more Unix!)
Lab 2- Scratch

• Basic way to get programming without being confused about syntax

• First time we learned about loops, if-conditionals, and object oriented programming
Lab 3- JavaScript

• Discussed the difference between HTML and JavaScript

• Learned how to do popups

• Learned about the basics of JavaScript (refer to JavaScript section)
Lab 4- Algorithms in JavaScript

- Learned how to call functions (more in JavaScript section)
- Learned how to access values held in HTML elements (more in JavaScript section)
- Binary search vs. random guessing
Lab 4- Binary Search

• Take our Battleships example

• Imagine we have a sorted list in ascending order

• We have picked one number

• The computer is trying to guess our number
Lab 4- Binary Search

List of Numbers: 1 2 3 4 5 6 7 8 9

My number: 9  
Computer guesses: 5

My reply: Number is higher
Lab 4- Binary Search

List of Numbers:  6  7  8  9

My number: 9

Computer guesses: 7

My reply: Number is higher
Lab 4- Binary Search

List of Numbers: 8 9

My number: 9
Computer guesses: 8

My reply: Number is higher
Lab 4- Binary Search

List of Numbers: 9

My number: 9

Computer guesses: 9

My reply: Right!
Lab 4- Binary Search

• Binary search is useful because we can “throw away” half of our list each time we make a guess

• The probability of getting a right answer based on the list increases with each guess

• The probability of getting a right answer based on random guesses never changes from turn to turn
Lab 5- Art Lab

- Two versions of this lab:
  - GIMP
  - Processing
- Meant as a way to show how computers (and code) can help with creating art
Lab 6- Eliza

• Meant to give you an idea of what artificial intelligence is like in computers (refer to lecture notes for more info!)
Lab 7- Hardware

- Introduction to logic and circuits
- Learned about a MUX
- Learned about testing two circuits for equivalency
Lab 7- Hardware

- How do I start to figure out what this circuit is doing?
Lab 7- Hardware

• See if you can find a pattern in how the inputs affects the output

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>select</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Lab 7- Hardware

Combinational Analysis

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Table</th>
<th>Expression</th>
<th>Minimized</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>select</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Lab 7- Hardware
Lab 7- Hardware

• When select is 0, input A seems to affect the output while input B seems to have no effect

• When select is 1, input B seems to affect the output while input A seems to have no effect

• **Conclusion**: the select input determines if input A or input B affects the output
Lab 7- Hardware

• How do we know if two circuits are equal?
Lab 7- Hardware

• Use a truth table to see if for every combination of inputs, you are getting the same output

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Decimal

Numbers can be from 0-9
Decimal

- Thousands: $10^3$
- Hundreds: $10^2$
- Tens: $10^1$
- Ones: $10^0$
Decimal

\[
\begin{align*}
\text{Thousands} & \quad 2 \quad \uparrow \\
10^3 & \\
\text{Hundreds} & \quad 4 \quad \uparrow \\
10^2 & \\
\text{Tens} & \quad 1 \quad \uparrow \\
10^1 & \\
\text{Ones} & \quad 3 \quad \uparrow \\
10^0 & \\
\end{align*}
\]

\[
= 2 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 3 \times 10^0
\]

\[
= 2413
\]
Binary

Numbers can 0 or 1
Binary

"Thousands" $2^3$

"Hundreds" $2^2$

"Tens" $2^1$

"Ones" $2^0$
Binary to Decimal

\[
\begin{align*}
1 & \quad 0 & \quad 1 & \quad 1 \\
\text{“Thousands”} & \quad \text{“Hundreds”} & \quad \text{“Tens”} & \quad \text{“Ones”} \\
2^3 & \quad 2^2 & \quad 2^1 & \quad 2^0
\end{align*}
\]

\[
= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0
\]

\[
= 11 \quad \text{← this is the decimal value where 1011 is in binary}
\]
Decimal to Binary

**Decimal**: 53

- What is the largest power of 2 that can fit into 53?
- \(2^5 = 32\)

Decimal value left to deal with: \(53 - 32 = 21\)

**Answer:**

<table>
<thead>
<tr>
<th>(2^5)</th>
<th>(2^4)</th>
<th>(2^3)</th>
<th>(2^2)</th>
<th>(2^1)</th>
<th>(2^0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Decimal to Binary

Decimal: 21

- What is the largest power of 2 that can fit into 21?
- \(2^4 = 16\)

Decimal value left to deal with: \(21 - 16 = 5\)

Answer:

\[
\begin{array}{cccccccc}
& & & & & & & \\
1 & 1 & & & & & \\
2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & \\
\end{array}
\]
Decimal to Binary

Decimal: 5

• What is the largest power of 2 that can fit into 5?
• \(2^2 = 4\)

Decimal value left to deal with: \(5 - 4 = 1\)

Answer:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
\end{array}
\]
Decimal to Binary

Decimal: 1

- What is the largest power of 2 that can fit into 1?
- $2^0 = 1$

Decimal value left to deal with: $1 - 1 = 0$

Answer:

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
</tr>
</tbody>
</table>
Decimal to Binary

Fill in the blanks with 0s.

Answer:

\[
\begin{array}{ccccccc}
1 & 1 & 0 & 1 & 0 & 1 \\
2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
\end{array}
\]
Hexadecimal

• Base 16

• Each digit can go from 0 to F (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

• Commonly represented by 0x __some_number__ (e.g., 0x2734)
Hexadecimal

Numbers can 0 to F
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)
Hexadecimal

“Thousands” $16^3$

“Hundreds” $16^2$

“Tens” $16^1$

“Ones” $16^0$
Hexadecimal to Decimal

\[ A \times 16^3 + 9 \times 16^2 + 2 \times 16^1 + 14 \times 16^0 = 43310 \]

This is the decimal value where A92E is in hexadecimal.
Decimal to Hexadecimal

**Decimal**: 53

- First, convert 53 to binary.
- Group \((110101)_2\) into fours from right to left

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
2^7 & 2^6 & 2^5 & 2^4 \\
0 & 1 & 0 & 1 \\
2^3 & 2^2 & 2^1 & 2^0 \\
\end{array}
\]
Decimal to Hexadecimal

**Decimal**: 53

- For each of the groups, pretend it is just an individual four digit binary number and find its decimal value

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>

$3 \equiv \text{II}$

$5 \equiv \text{II}$
Decimal to Hexadecimal

Answer:

\[
\begin{align*}
0 \times 3 & \quad \quad \quad 5 \\
"Tens" & \quad \quad \quad "Ones" \\
16^1 & \quad \quad \quad 16^0
\end{align*}
\]

We can double check by \(3 \times 16^1 + 5 \times 16^0 = 48 + 5 = 53\)
HTML/CSS

• HTML is used to create webpages

• CSS is used to style the webpages (it makes your webpage pretty!)

• HTML can exist without CSS

  • There are default fonts/sizes for headings that will be used if no CSS is used
• Tags are enclosed in <>
  • Usually in pairs with one open tag and one closing tag (e.g., <h1> and </h1>)
• Certain tags are mandatory in order to have your webpage display
HTML

<html>
  <head>
    .
    .
  </head>

  <body>
    .
    .
    .

  </body>
</html>
HTML

- Different tags do different things
- `<hr />` will draw a line across the page
- `<br />` is a line break
- `<img src="...">` will put the image from the link inside the quotation marks on the page
- `<a href="...">text</a>` will make the text between `<a>` and `</a>` a hyperlink
CSS

- CSS can be incorporated with your HTML in three different ways
  - Separate stylesheet
  - Inside `<style>` tags in your HTML
  - Inside individual tags (refer to midterm 1)
- Uses American spelling
CSS

• Put all your CSS in a separate stylesheet and reference that stylesheet from the HTML file

• This makes your code much cleaner and easier to read

  <head>
    <link rel="stylesheet" type="text/css" href="nameOfStylesheet.css">
  </head>
CSS

• Like midterm 1 problem 3

```html
<head>
  <style>
    h1 {
      color: purple;
      text-align: center;
    }
  
  </style>
</head>
```
CSS

- Inside individual tags

  <h1 style="color: purple; text-align: center">
JavaScript

- JavaScript gives functionality to your HTML

- HTML is what is displayed on the page (e.g., buttons or text fields) but JavaScript is what makes the button do something

- Two ways to include JavaScript in your HTML file
  - Inside `<script>` tags
  - In a JavaScript file that the HTML references
JavaScript Variables

• Variables are used to hold values

• These values can change when you need them to

  • Variables are not limited to only holding one value
JavaScript Variables

• Variables need to be declared before it can be used

• Declare using:
  
  • `var `nameOfVariable`;  

• If you do not give the variable a value during declaration, it will hold a value of undefined

• Give a variable the value of Hello World! by:
  
  • `var `nameOfVariable` = “Hello World!”;`
Scope of Variables

• Variables don’t exist forever

• How long they last for depends on where they are declared
  • Within a loop vs. within a method vs. in the class
Scope of Variables in a Loop

```javascript
var someNumbers = [1, 2, 3, 4, 5];

function sumList(list) {
    var sum = 0;
    for(var i = 0; i < list.length; i++) {
        var curValue = list[i];
        sum = sum + curValue;
    }
    return sum;
}
```

How long do you think i will last for? What about curValue, sum, and list?
Scope of Variables in a Loop

• Variables declared **inside** a loop only last for that iteration

```javascript
for(var i = 0; i < list.length; i++) {
    var curValue = list[i];
    sum = sum + curValue;
}
```

*curValue is deleted after the iteration finishes and recreated when the next iteration starts. You can’t use this variable to hold a value to use during the next iteration!*
Scope of Variables in a Loop

• Variables declared **inside a loop condition** only last until that loop ends

```javascript
for(var i = 0; i < list.length; i++) {
    var curValue = list[i];
    sum = sum + curValue;
}
```

You would not be able to access the value of `i` after the loop has finished.
Scope of Variables in a Function

```javascript
function sumList(list) {
    var sum = 0;

    for(var i = 0; i < list.length; i++) {
        var curValue = list[i];
        sum = sum + curValue;
    }

    return sum;
}
```
Scope of Variables in a Function

- Variables declared *in a function* only last until that function ends (i.e., local variables)

- If you try to use that variable outside of the function, you will get an error!

- This does not mean that you cannot have local variables with the same name in different functions.
Scope of Variables in a Function

```javascript
function sumList(list) {
    var returnValue = 0;
    for(var i = 0; i < list.length; i++) {
        returnValue += list[i];
    }
    return returnValue;
}
```
Scope of Variables in a Function

```javascript
function averageList(list) {
    var returnValue = 0;

    returnValue = sumList(list)/list.length;

    return returnValue;
}
```
Scope of Variables in a Function

- Having `averageList` and `sumList` both use a local variable called `returnValue` is fine.
- Don’t expect there to be a `returnValue` variable holding the value of `averageList` or `sumList` outside of these functions.
Scope of Global Variables

```javascript
var someNumbers = [1, 2, 3, 4, 5];

function sumList(list) {
    var sum = 0;
    for(var i = 0; i < list.length; i++) {
        var curValue = list[i];
        sum = sum + curValue;
    }
    return sum;
}
```
Scope of Global Variables

• Variables declared **globally** (outside of any function) lasts until that program ends

• Any function can use these kinds of variables

• Values assigned to these variables will be held even after a function ends

• Declaring too many will cause a memory overflow!
JavaScript Functions

• Functions are a set of instructions packaged into one

• Similar to writing sections of an instruction manual

• If someone needs to know what to do when a certain situation occurs, they can look up that section of the instruction manual
JavaScript Functions

- Declare a function by:
  - `function functionName() {
    (insert code to determine what you want the function to do)
  }

- Functions can have parameters
JavaScript Functions

• If I wanted to find the sum of all the numbers in the list held by the variable someNumbers, I would need to call sumList like this:

    sumList(someNumbers);
JavaScript Function Parameters

• Parameters are values you can pass into the function

• Parameters tend to be pieces of information that the function needs in order to do its job

• E.g., If you want to calculate the average of a number, you need to know what number to calculate with. A function can get this number by being the number as a parameter.
JavaScript Return Values

• For example:

```javascript
function sumList(list) {
    var sum = 0;

    for(var i = 0; i < list.length; i++) {
        var curValue = list[i];
        sum = sum + curValue;
    }

    return sum;
}
```
JavaScript Return Values

• Return values are values that a function can produce and give back to whoever called it.

• E.g., If someone called a function to calculate the average of a number, he/she would want to know the answer at the end of the calculation. A return value would give that answer to the user.
JavaScript Return Values

• A return value can be declared by:

  • function functionName() {
      (insert code to determine what you want the function to do)
      return whatYouWantToReturn;
  }
JavaScript Return Values

- For example:

  ```javascript
  function sumList(list) {
    var sum = 0;

    for(var i = 0; i < list.length; i++) {
      var curValue = list[i];
      sum = sum + curValue;
    }

    return sum;
  }
  ```
JavaScript Functions

• To use a function, we need to call it

• We can call it using the function’s name along with any parameters it needs

• Even if a function has no parameters, it still needs to have a () after the function name in order for the compiler to know it is a function and not a variable
JavaScript Functions

var someNumbers = [1, 2, 3, 4, 5];

function sumList(list) {
    var sum = 0;

    for(var i = 0; i < list.length; i++) {
        var curValue = list[i];
        sum = sum + curValue;
    }

    return sum;
}
JavaScript Events

• Events are predefined JavaScript functions (or lines of code) that occur when something happens.

• E.g., If a button is clicked, an alert will popup. The alert is the event.

• Defined using things like onClick, onMouseOver, onMouseOut, etc.
JavaScript Control Flow

• if conditionals are a way to choose what will happen

• Usually take the form of:

  • if(something happens or something is true) {
    something happens
  } else {
    something else happens
  }
function checkNumber(number) {
    var evenOrOdd = " ";

    if(number % 2 == 0) {
        evenOrOdd = "even";
    } else {
        evenOrOdd = "odd";
    }

    return evenOrOdd;
}
JavaScript Control Flow

- checkNumber only returns "even" if the number is divisible by 2
JavaScript Control Flow

• Loops are a simple way to repeatedly run a few lines of code for however many times you want

• Each time the loop is run is called an iteration

• Two types of loops:
  • For loop
  • While loop
For Loops

• Run a loop for a predetermined number of times

• Each time the loop is run is called an iteration

• Declared like:

  • for(var i = 0; i < list.length; i++) {
    do something
  }

For Loops

var list = [1, 2, 3, 4, 5]
var total = 0;

for(var i = 0; i < list.length; i++) {
    total = total + list[i]
}

What we start with.
This is the value i has before the for-loop has ever been run.
For Loops

var list = [1, 2, 3, 4, 5]
var total = 0;

for(var i = 0; i < list.length; i++) {
  total = total + list[i]
}

End loop condition (determines what has to be untrue in order for the for-loop to end). You can also think of it as how many times the loop will run for. This condition is checked at the beginning of each iteration.
For Loops

var list = [1, 2, 3, 4, 5]
var total = 0;

for(var i = 0; i < list.length; i++) {
    total = total + list[i]
}

How much i will change by after the for-loop has finished each iteration (could also be i+2, i--).
This step is run at the end of an iteration.
For Loops

```javascript
var list = [1, 2, 3, 4, 5]
var total = 0;

for(var i = 0; i < list.length; i++) {
    total = total + list[i]
}
```

Will be executed during each iteration of the for loop (in this case, it will be executed 5 times).
Putting It All Together

• Midterm 2 Problem 3:

Complete the following function that takes a number “n”, and returns the average of the numbers from 1 to n.

Example: averageN(5) returns \( \frac{1+2+3+4+5}{5} = 3 \)
Putting It All Together

• Midterm 2 Problem 3:

Complete the following function that takes a number “n”, and returns the average of the numbers from 1 to n.

Example: \texttt{averageN}(5) \text{ returns } \frac{1+2+3+4+5}{5} = 3

\begin{itemize}
  \item Tells us the name of the function is \texttt{averageN}
\end{itemize}
Putting It All Together

• Midterm 2 Problem 3:

Complete the following function that takes a number “n”, and returns the average of the numbers from 1 to n.

Example: averageN(5) returns \( \frac{1+2+3+4+5}{5} = 3 \)
Midterm 2 Problem 3:

Complete the following function that takes a number “n”, and returns the average of the numbers from 1 to n. 

Example: averageN(5) returns (1+2+3+4+5)/5 = 3
• Midterm 2 Problem 3:

Complete the following function that takes a number “n”, and **returns the average** of the numbers from 1 to n.

Example: `averageN(5)` returns \((1+2+3+4+5)/5 = 3\)
Putting It All Together

```javascript
function averageN(n) {
    var sum;

    for(var i = 0; i <= n; i++) {
        ???????
    }

    return _____;
}
```

This is what we know from the question. Now, let’s decide what the function needs to do.
function `averageN(n)` {
    var sum;

    for(var i = 0; i <= n; i++) {
        sum += i;
    }

    return `____?____`;  // Sum is now the value of 1 + 2 + … + n. What needs to be done to sum in order to make it an average?
Putting It All Together

function averageN(n) {
    var sum;

    for(var i = 0; i <= n; i++) {
        sum += i;
    }

    return sum/n;
}

Divide it by n!
Putting It All Together

```javascript
function averageN(n) {
  var sum;

  for(var i = 0; i <= n; i++) {
    sum += i;
  }

  sum = sum/n;

  return sum;
}
```

You could also do the division before returning `sum`. 
Common JavaScript Errors

- Wrong spelling (e.g., variable name spelled wrong, method name spelled wrong)
- Case sensitivity
- Forgot to match {}, (), []
  - for every { that appears, there should be a }
- Index for arrays start at 0, not 1!
Common JavaScript Errors

• Did not give a variable a value before trying to ask it for a value

• Did not declare a variable prior to using it

• Didn’t call functions properly

• Variable scope
Q&A

• Questions?