|  |
| --- |
| **CS 164 Introduction to Computer Science Fall 2013**  **Study Guide for Midterm Exam 2 Monday, November 18, 2013 8:00 - 8:50 AM Nesbitt 111** |

From the Lectures:

* Overview of HCI - example of HCI problems (Human Computer Interactions)
  + “Human-Computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.”
  + Types of majors included in HCI:
    - ***Computer Science*** - designing applications and user interfaces.
    - ***Psychology*** - applying theories of human behavior
    - ***Sociology and Anthropology*** - studying the interaction between technology, work and organization.
    - ***Engineering*** - designing products
  + Examples of HCI:
    - ***Interface Design***: the graphical user interface, windows, frames, application interfaces
    - ***Interface Hardware***: the mouse, touch screens, voice recognition, devices for the disabled
    - ***Ergonomics***: repetitive motion disorder, lower back pain, eye strain, fatigue, lighting issues, special needs (left handedness, color blindness, interfaces for children, etc.)
    - ***Embedded Computers***: microwaves, cell phones, cars, entertainment systems
    - ***Human Learning***: educational software, on-line help, information kiosks
    - ***Communication***: e-mail, instant messaging, group interfaces
    - ***Adaptation***: users adapting to machines, machines that adapt to users
    - ***Computer Timing***: does the computer respond too slow? or too fast?
    - ***Social Implications***: consequences of improved productivity demands, computers replacing humans in the work force, cool colors for my PC
* Interface Design Principles - Transparent, Forgiving, Visual, Intuitive
  + A good user interface is:
    - ***Transparent*** - the user of the software/computer should not need a technical background to operate it. Technical issues should be managed by the computer, and hidden from the human.
    - ***Forgiving*** - the software should allow the user to undo the most recent action. Particularly devastating actions ("Delete this file?") should be prompted for confirmation.
    - ***Visual*** - sight is the primary sense for most people. The interface should use this medium as well.
    - ***Intuitive*** (not in text) - very few people read manuals before using a piece of software. A good interface can be "figured out" without resorting to the instructions.
* Highlights from "The Macintosh User Interface Guide"
  + The *user should control the computer* (not the other way around).
  + The computer should be *responsive*: the user can perform a task quickly and directly.
  + The computer should be *permissive*: the user can do "anything reasonable" in the program. Error messages should be minimized.
  + The computer should be *consistent*: menus on different applications should be similar (**Open** and **Save** should always been under **File**, use **ctrl-c**, **ctrl-v** and **ctrl-x** for cut and paste, etc.).
  + Frequently used commands should be at the top of a menu.
  + Dangerous commands should be at the bottom of a menu.
  + Options (in dialogue boxes, etc.) should always default to the safest option.
* JavaScript -
  + what is it? (what it's not i.e. Java)
    - JavaScript is a scripting language for client and server applications.
    - JavaScript is object oriented and event driven.
    - JavaScript commands may be embedded in HTML documents and are interpreted by the client.
  + **Script**
    - <script> is used to place JavaScript in an HTML document. A document can have multiple script tags.
  + Embedding and hiding it in HTML
    - <script> tags,
    - specifying a file as the JavaScript source,
    - specifying a JavaScript expression as the value for an HTML attribute,
    - as event handlers within certain other HTML tags (mostly form elements).
  + comments in JavaScript vs. comments in HTML
    - <!-- html comments --> //javascript comment
  + using document.write()
    - document.write() displays text and HTML tags, values of variables, and can also handle multiple outputs on a web page. We use “+” to concatenate strings or variables.
    - document.write("Here's some text.")
  + Intrinsic JavaScript Objects
    - Date, Math, Navigator, Document, and String
  + document object
    - properties: (document.referrer, document.lastModified, ...)
    - methods: (document.write(), document.title())
  + arithmetic expressions, assignment statements
    - =, +, -, \*, /, %, ++(increment), --(decrement)
  + string expressions
    - anchor(name)
    - big() – makes string big
    - blink()
    - bold()
    - charAt(number)
    - fixed() – teletype text
    - fontcolor(color)
    - fontsize(size)
    - indexOf(text, index) - Locates the position of text in the string. Starts looking at position index. Returns -1 if text is not found.
    - italics()
    - lastIndexOf(text, index) - Locates the last position of text in the string. Starts looking at position index. Returns -1 if text is not found.
    - link(URL)
    - small()
    - split(separator) - Divides the text into an array of strings.
    - strike()
    - sub() - Text is subscripted
    - substr(first, n) - Extracts n characters starting with first to make a new string. (See examples below). Strings begin at position 0.
    - substr(first) - Extracts characters starting with first and continuing to the end of the string to make a new string. (See examples below). Strings begin at position 0.
    - substring(first, last) - Extracts the characters between first and (up to but not including) last to make a new string. (See examples below). Strings begin at position 0.
    - sup()
    - toLowerCase()
    - toUpperCase()
  + conditional expressions, if statements, if/else statements
    - used to alter the Flow of Control.
  + JavaScript functions (making, using, parameter passing)
    - a collection of JavaScript code that can be placed in the <head>...</head> portion of the HTML document for use in another location.
    - function funcName(parameter list)
    - { ... JavaScript Commands ... }
  + communication between forms and functions
    - Functions are often executed on a cue from an event handler, called by a form
  + loops
    - a set of commands that executes repeatedly until a specified condition is met.
    - For, do while, while
* The Object Oriented Paradigm
  + Origins
    - based upon theories from the field of cognitive science about how information is represented in the human mind.
  + Advantages and Disadvantages
    - Disadvantages: Size, Effort, Speed
    - Advantages: Code Reuse and Recycling, Encapsulation, Design Benefits, Software Maintenance
  + Attributes and Methods
    - This data helps you identify what the object is, and distinguish it from other objects of it's type.
  + Object Oriented nomenclature in Computer Science (i.e. "who.what")
    - If we want to refer to an object we use its name, if we want to refer to one of its attributes, we put a “.” Between the two.
* HTML - tags, forms, event handlers
  + Tags: <html>, <head>, <body>, <title>, <bold>, <i>, <u>, <tt>, <hr>, <br>, <p>
    - Attributes: FACE, BGCOLOR, TEXT, COLOR, SIZE, ALIGN, WIDTH
  + A form is formatted document containing blank fields that users can fill in with data
    - In HTML, the following kinds of interaction are allowed when using forms:
    - Single line typed input,
    - Multiple line typed input,
    - Simple buttons, Radio buttons,
    - Pop-up menus with single or multiple selections,
    - File submission.
  + onAbort, onBlur, onChance, onClick, onError, onMouseOver, onMouseOut, onLoad, onSubmit
* Binary Numbers, Binary Addition, Binary Subtraction
  + In a base two or binary number system, the digits from left to right represent greater powers of two
  + Addition in base two works just like addition in base ten. The only difference is that 1 + 1 = 10. Same with subtraction, reversed.
* Computer Representation of Integers (including two's complement for negative integers)
  + Most computers store integers in either 4 byte or 8 byte blocks called the system word. The two’s complement is obtained by flipping all the 0’s and 1’s.
* Computer Representation of Characters (ASCII)
  + All character data must be represented on a computer as collections of bits. American Standard Code for Information Interchange. In the ASCII character set, each binary value between 0 and 127 is assigned to a specific character.
* Binary Fractions
  + bits to the right the radix point stand for halves, quarters, eighths, etc. in base 2: (multiply by 2 and check to see if it’s over one, use remainder to continue)
* Computer Representation of Floating Point Numbers (exponent, mantissa, etc.)
  + **s eee....e mmm....m**
  + (1+f)\*2e
* Errors arising from representation of integers and floating point numbers
  + An error will occur if the number is truncated because of overflow. An integer cannot always be computer in the number of digits. This also goes for floating point numbers.
* Computer Hardware:
  + Basic Gates: AND, OR, NOT

AND OR NOT

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0 | 1 |
| 1 | 0 |

* Hexadecimal Numbers
  + Hexadecimal functions on base 16. 16^n…16^2 \* (1st digit) + 16^1 \* (second digit) + 16^0 \* (third digit)… also digits run from 1-16 with A-F starting at the 10th place.
* HTML colors
  + #FFFFFF, #888888, #000000
* ASCII, Unicode
  + All character data must be represented on a computer as collections of bits. The most commonly used system (in this country) is ASCII(American Standard Code for Information Interchange).

From the Reed Text:

* Chapter 4: JavaScript and Dynamic Web Pages
  + JavaScript objects (intrinsic objects, using objects to access form elements)
* Chapter 5: JavaScript Numbers and Expressions
  + Mathematical formulas and calculations in JavaScript
* Chapter 7: Event-Driven Pages
  + HTML forms (creation and use)
* Chapter 9: Abstraction and User-Defined Functions
  + JavaScript functions (creation and use)
* Chapter 11: Conditional Execution
  + if statements, if/else statements, if/else if/else... statements
  + loops (while statements)
* Chapter 12: Data Representation
  + Binary and Hexadecimal
  + Floating-Point Representation
* Chapter 14: Inside the Computer - The von Neumann Architecture
  + *Knobs & Switches* simulator
  + Basic Assembly Language Instructions (LOAD, STORE, ADD, SUB, NOP, HALT)
  + *no need to memorize Machine Language Instructions*
* Chapter 15: JavaScript Strings
* Chapter 16: Inside the Computer: Transistors and Integrated Circuits
  + Truth Tables
  + Basic Gates (AND, OR, NOT)
  + Logic Circuits

From the Labs:

* JavaScript!!!
* HTML forms!
* Binary representation.
  + I’m fine on this.

**Types of Problems to Expect:**

1. You can expect to see short answer, fill in the blanks, TRUE/FALSE, and multiple choice questions
2. You can expect to see questions involving key terms and nomenclature from all areas of study
3. You can expect to see questions similar to those in the text on *number systems* and *program translation*.
4. You can expect to see problems dealing with various aspects of your homework assignments and labs.
5. You can expect to see problems of the form "What appears in the browser window when the following HTML code appears on a web page" and "What HTML code should appear on a web page in order to produce the following output in the browser window"
6. You can expect to see problems of the same form for JavaScript
7. You can expect to see problems that involve communication between HTML forms and JavaScript scripts.