Notes. Volunteers?

https://bit.ly/**pui-lecturenotes**

Midterm review

Programming Usable Interfaces

05-430/630 Fall 2023

Human-Computer Interaction Institute

Alexandra Ion

Anthony Levin-Decanini

Carnegie Mellon University

1 Logistics

2 Recap of material

3 Study tips

Heads-up: Midterm Quiz

Next **Monday, Oct. 9**

Topics include everything in class so far, including today's lecture

- all lectures, labs, required readings
- labs: including Lab 5 and HW4 topics

“Traditional” Procedure:

- in-person, on paper, **closed book**
- HERE, in **TEP 1403**, 8:00am
- **60 min** (timed quiz)

Instructions

- **bring your CMU ID!**

- leave your backpack, jackets, etc. outside the lecture hall
- bring non-erasable pens

PUI Midterm Quiz

05-430/630 Programming Usable Interfaces, Fall 2023
Human-Computer Interaction Institute, CMU.
Wednesday, October 9, 2023

Your NAME: _____ Number of pages handed in: _____

Section: ( ) **A** w/ Yi Fei ( ) **B** w/ Howie ( ) **C** w/ Yi-Hao ( ) **D** w/ Pranav ( ) **E** w/ Seyun

Instructions

Do this first: add your **name** on **every single page**, as we will distribute the sheets for grading. If you need to use extra sheets, remember to add your andrew ID there too.

This is a *closed book* exam. No materials whatsoever may be used.

Make clear what we should grade. Make sure your *writing is legible* and your *markings are unambiguous*. Only have **one answer** for us to grade, i.e. strike out other versions, make it clear what we should grade. **No erasable pens** (e.g., pencils, etc) allowed. Only answers written with non-erasable pens will be graded.

Enter the number of pages you handed in above. Make sure that you hand in all pages that we should grade, incl. any additional pages you might have used. Please **write only on the front pages**. Ask us for more paper, if you need.

Only one person may hand in at a time. Before getting up, check that no one is walking up to the front, otherwise please wait your turn. Then leave the room quietly. You might hand in early, up to 10 min before the exam ends. After that, please remain seated.

Tip: If you are stuck on a question, move on to the next. Do all "easy" questions first. Add a checkmark for each question you are sure of, this avoids that you have to parse questions over and over again.

We use the following visual conventions:

Radio buttons, i.e., make **only one** selection:

( ) option 1 ( ) option 2 ( ) option 3

Check boxes, i.e., multiple selections possible, guaranteed that **>1** to select:

[ ] option 1 [ ] option 2 [ ] option 3

Suggested Materials

Lecture & Lab materials

- All slides on Canvas: https://canvas.cmu.edu/courses/36440/files
- Lecture notes: https://bit.ly/pui-lecturenotes
  - Everything *discussed* in class (lecture & labs) is part of the material

HTML + CSS + JavaScript

- Code examples from lectures & labs: https://github.com/interactive-structures/pui-materials
- Review your homework

All required readings: We discussed a lot but not all of it in class

Tentative Quiz Structure

Makeup of the quiz

Expect it to be proportional to the contents taught so far:

~65% programming
~35% UX

Types of questions

- True/False questions
- Multiple choice & Single choice questions (marked as such)
- Few free text, short answer questions
- Programming questions, for example
  - what does this code do
  - what is wrong
  - code few lines to do X
  - etc.

Topics in this slide deck are not comprehensive.

You need to review **all** materials we covered until today for the midterm.

Please don’t ask if X will be on the exam.

The answer will be YES.

1 Logistics

2 Recap of material

3 Study tips

HTML

- Contrast the main responsibilities of HTML, CSS, and JavaScript
- What HTML stands for
- What is hypertext, might ask you to sketch it
- What are HTML tags & attributes
- How is HTML parsed
- Grouping elements: why?
- Container elements: block-level vs inline
- What semantic elements are, their benefit(s) over the older, generic elements
- Good code style: Blank Lines and Indentation, Use Lowercase Element & attribute Names, Close All HTML Elements, ...

HTML: example questions

- Give one example for an inline element
- We give you short HTML code, ask you to identify the tag and the attributes
- Give one example of a semantic element that can replace a `<div>`
- We give you HTML code and ask you to draw the visual output

CSS

- Describe the main role of CSS within web technology
- What is a CSS declaration, property, selector
- Selectors: element, class, ID selector → how to specify them in HTML & CSS
  - Combining selectors with space, e.g., `p.big li {}`
  - Grouping selectors with `,`
  - What pseudo-class selectors are, how to use them, give one example
- Methods of placing CSS in your HTML code, know which one is better/best and why
  - external file, style tag, inline style
- CSS rules for resolving conflicts:
  - origin precedence rule, inheritance rule, specificity rule
- Layout models: box model, flexbox

CSS: example questions

- Example code with different selectors, describe or identify how the content will look (color, border, background, …)
- Write a CSS declaration that changes all `p` content text color to red
- Given an example CSS box with content, padding, border specifications, draw & label the new box after setting `box-sizing: border-box`
- Given HTML content and CSS code with flexbox declarations, draw the layout as it will be rendered

Look at the in-class code examples (lectures & labs) and the homework

JavaScript: DOM

- Understand what the DOM is
- How you can dynamically change a webpage using JS
- The DOM represents the HTML tree structure
- Select elements from the DOM: `document.querySelector()` using selectors to retrieve a reference to the first match
- Can add and remove elements
- Can change element contents and element attributes, incl. CSS classes and IDs

JavaScript: Basics

- You can include JS everywhere, understand the differences and what is best coding style (think of `defer`)
- Variable declaration, naming, types
- Conditionals, Booleans and equality operators
- Arrays: indexing, retrieve element, length
- Purpose & structure of loops (for loop, for-of loop)

JavaScript: Advanced

- What are functions, how to declare them (syntax!), use of parameters and return values
- Scope: objects exist in the block they are declared in, not where they are called
- Events: what events are, how you add event listeners, how to declare an event handler/callback function
- Object literal: made up of multiple members, each has a name and a value. A member can be any object, e.g., variable, array, object, function, …
- Constructors: how to define them, what happens when you call one

JavaScript: example questions

- Given HTML structure, draw the DOM tree
- Name the in-built JavaScript object that enables you to access the DOM
- Given HTML structure, write the JS code to select a highlighted element
- Combine two given strings into one
- Name 3 different event types
- Given an array, retrieve a given element
- Add a given text to a given paragraph element

Algorithmic thinking

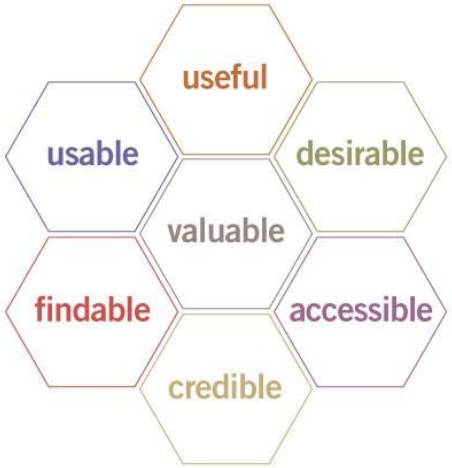
- The steps of algorithmic thinking
  1. Understanding the problem: What is input, what is output?
  2. Formulating the problem: data structure, architecture
  3. Developing the Algorithm: define functions, code blocks, program flow
  4. Implementing the Algorithm: make test cases that you can run repeatedly
  5. Running it on the data (if applicable)
- Types of errors: syntax, logic errors
- Debugging & programming tips: implement small parts first and test frequently, print variable values in the console, read the error message, ...
- Code repetition is the root of all evil in software

Algorithmic thinking: example questions

- Given a problem description, break it down into small tasks that can be converted into simple code (e.g., variable assignment/increment, conditional, loop, etc.)
- Describe the difference between logic and syntax error
- Given some short code, find the error
- Describe specific steps for debugging
- Order given steps of debugging from what to do first to last

Usability Basics

- Relationship between CX, UX, and UI and examples of each
- Elements of every UX: Content, Context, and Users
- Facets of UX ▸
- Usability: definition & components
- Why usability matters
  1. Life & death
  2. To businesses
  3. To product teams



Accessibility

- Why accessibility matters
- What are the types of disabilities, what are the models of disbility
- What is accessibility
- Assistive technologies, give examples (physical & digital)
- Web accessibility:
  - Visual: screen readers, adjust contrast/color schemes
  - Auditory: Captions, Sign Language interpretation
  - Motor: Eye trackers, adaptive buttons
  - Cognitive: Simple language, consistent navigation, color choice
- Web content accessibility guidelines: 4 principles
- Tips for text, selection, keyboard access

Accessibility: example questions

- Give an example for assistive technology for motor disability
- Give an example of accessibility for cognitive disability
- Name at least two ways of improving web accessibility.
- Give one example for how to satisfy the WCAG principle Robust on level A.
- Give examples of good alt text.
- Improve bad alt text.
- Make a case for why accessibility is important, incl. evidence.

Human Factors: Sensation & Perception

- Origin of the field of human factors
- Dreyfuss' *The Measure of Man*
- Senses
- Visual system
  1. Anatomy
  2. How we sense motion, color, contrast
- Auditory system
  1. Anatomy
  2. How sound is processed in the brain
- Perception & design implications:
  1. Color
  2. Contrast
  3. Legibility
  4. Visual field, tracking & scanning
  5. Visual attention

Human Factors: Cognition

- Historical vs. contemporary models of cognition
- Affordances & Signifiers: definitions & examples
- Conceptual models: definition & examples
- Feedback: definition & examples
- The role of emotion in cognition
- Memory:
  - Dual process theory
  - Short term vs long-term
- Mapping & grouping
  - Gestalt principles and grouping
- Cognitive biases

1 Logistics

2 Recap of material

3 Study tips

Suggestions for Studying

Review the lecture slides, lecture notes & lab notes

Re-read the required readings

HTML/CSS/JavaScript

- Go through all the exercises (code) from the labs & lectures

Talk with other students in class about what you're learning

- What (you/they) thought were key ideas + concepts
- Asking them to describe those ideas + concepts (or vice versa)

Study tips

1. Read your lecture notes
    - Highlight or make marginal notes for important words or concepts. This will help fix ideas and will help you to actively learn the material.
    - Re-do examples yourself, step by step. Examples often look easy when explained in class, but often turn out to be much harder when you do them yourself.
    - Write down questions about things you do not understand. Bring these questions to lecture, lab, and to office hours and ask them.
2. Re-read the required readings. As you read, highlight, re-work examples yourself, and write down questions, as suggested above.
3. Review your homework assignments, solutions and feedback. Event if you did not lose points on a homework question does not necessarily mean you got everything right.

end.