CS 97 Midterm Study Guide

This exam will most likely have a roughly even mix of coding questions, short-answer questions, and long-answer questions.

The midterm will be uploaded to CCLE in PDF format. It will have space for answers. Whether you do it digitally or on paper (either printing out the PDF or just using blank paper with question numbers), **make sure you have the necessary apps and/or materials ready beforehand**.

For the midterm, it is assumed that you've done Homework 1 and have *read* Homework 2. Everything talked about in lecture is fair game. However, you will of course be more heavily tested on topics that more time was spent on. Here is the complete list of topics in **relative** order (in the humble opinion of the LAs) of importance. You should spend the majority of your study time getting a solid foundation on the items on the first-half or so of the list :

- Files (objects) and processes (actions)
    - Emacs and processes
    - Emacs is a process
    - Getting around in Emacs, e.g.
        - `C-h k`
        - `C-x o`
        - `C-x C-c`
        - `C-g`
        - `C-x 1, C-x 2, C-x k`
        - `C-s`
        - `C-x C-f`
        - `C-v, M-v`
        - `M-x man`

    - `M-x shell`; what is a ”shell”?
- Emacs Lisp
    - C-j etc.
    - Defining and using functions
    - Lists and other objects
    - Quoting
- The shell
    - Variables
    - Pipes, redirection
- POSIX, portability and standardization
- POSIX commands
    - `cat, ls, touch, rm, rmdir, cp, echo`, etc.
    - `pwd`
    - `mknod`
    - `grep`
- Regular expressions
    - BRE vs ERE syntax
    - Everything you can look up but here’s a handy cheat sheet:

      http://web.mit.edu/hackl/www/lab/turkshop/slides/regex-cheatsheet.pdf
    - Character encoding
        - `locale`
        - ASCII
        - UTF-8
- POSIX Filesystem
    - regular files
    - directories
    - hard links

    - symlinks
    - file name resolution
    - special files
- Python
    - Basic syntax
        - Variables
        - Control Flow
            - if-elif-else statements
            - for/while loops
            - try/except blocks
        - Functions
        - Classes
    - Objects basics
        - Classes are objects, too
        - Operator overloading (e.g., __add__)
    - Classic python builtin types (sequences, dictionaries, callables, etc.)
        - and operations on those types
    - Variants (CPython, JPython, etc.)
    - **Modules and Packages**
        - Motivation > mechanism
        - mechanisms for dealing with issues related to packaging: pip, venv
        - Dependencies, conditional dependencies
        - No file structure details
    - Namespace control; import
        - `if __name__ == "__main__"`
- Little languages and scripting languages

    - sh, Lisp, Python, JavaScript, etc.
    - Interpreters (e.g., Python VM, JVM)
- JavaScript, React
    - High level stuff before the midterm
    - Assumes an attempt at the React homework, basic understanding of the tutorial

Study and Test-taking Tips from the LAs

- **Breathe!** The material may seem quite comprehensive but focus on the basic concepts and principles rather than pedantic details, such as exact syntax or knowing every option of a certain shell command.
    - The midterm is **open computer**, if you don't know what a program does you can always Google it or use the man pages.
    - The most important thing to "learn" is where/how to find the answers to questions like that. ^
- **Make cheat sheets ahead of time.** You know best how you think, and when you make your own material you know exactly where what you're looking for is. Cheat sheets are also a great way to study!
- **When reviewing lectures, think about the big picture** - what trade-offs do software developers make when choosing between different languages? Why is it important to have good version control systems? How can you quickly learn how to use a specific POSIX program/Python library/etc. given that it's impossible to know everything?
- **Do not waste your time on the midterm checking to make sure code you write is exactly correct.** Chances are you will lose few points (if any at all) for small syntax errors or not handling an edge case.

- **There will be open-ended questions on the midterm where there is no right answer.** Always focus on the quality of your justifications (and writing concrete, specific facts) over whether or not you're "right".