CPSC 131 MIDTERM EXAM STUDY GUIDE SPRING 2023

When

- Wednesday/Thursday, March 22/23 during normal class time

Logistics

- In-person paper exam in your normal classroom and time
- No electronic devices allowed. You'll be asked to put everything in your backpack, including phones, and leave your backpack at the front of the room
- Be prepared to show photo ID. CSUF randomly sends someone to check IDs for major events, like midterm or final exams.

Format

- Closed-books, closed-notes. Absolutely no collaboration in any form with anyone else.
- Paper exam.
    - You'll be given a paper test, you'll write on the paper, and then return the paper to me. You may not bring your own paper.
    - Your submitted exam will be used to collect attendance that day – you do not need to check-in with iClicker
- There are four parts to the exam, each weighted roughly the same
    - Multiple choice, true/false, short answer
    - Data structure sketching
    - Analyzing alternatives essay
    - Coding
        - Be prepared to add and implement recursively a new member function to one of our existing abstract data type classes taken from our Implementation Examples.
        - Your implementation may require you to add an overloaded helper function as well.
        - Your code must be syntactically and semantically correct. Approach this as though you need to compile and run the code you write.

Topics Covered

- Part 0 - Introduction & Review
  - C++ pointers, references, arrays, dynamic memory, Object Oriented Programming (OOP), classes/structs, templates, exceptions
  - Writing/updating classes with proper
    - encapsulation (public/private)
    - instance attribute data members
    - constructors, destructors
    - overloaded queries, accessors, and mutators
    - overloaded operators
    - etc.
  - Algorithm complexity analysis
    - asymptotic analysis (big-O)
    - efficiency classes for the fundamental operations of all the data structures covered
      - $O(1)$, $O(\log_2 n)$, $O(n)$, and $O(n^2)$
    - choose a container by comparing efficiency classes of operations
  - Memory model (stack vs. heap)
  - Iterative and recursive algorithms
- Part 1 - Sequence Containers
  - Arrays & Vectors
    - Fixed & Extendable implementations
    - Amortized efficiency, complexity analysis
  - Lists
    - Singly & doubly linked lists
    - Null-terminated, circular with one dummy node
    - Complexity analysis
  - Concepts & Interfaces
- Part 2 – Iterators
  - Iterator Concepts & Interfaces
  - Pointers as iterators
  - Container Traversal Techniques
    - Iterative & recursive
- Part 3 - Container Adapters
  - Stacks, Queues
  - Concepts & Interfaces
  - Array, Vector, List implementations
  - Complexity analysis