CS106B
Spring 2016 Cynthia Lee

Final Exam Topics

Topics list adapted from one by Marty Stepp.

This handout is a list of topics to study for the final exam next week. During the later part of section, your section leader will answer your questions about topics for the final exam.

Midterm exam topics study list (fair game for final exam but less emphasis):

- **C++ basics**: strings, streams (file I/O), functions; passing by value and by reference
- **Using ADTs**: the collections from the Stanford C++ library, such as `Vector`, `Grid`, `Stack`, `Queue`, `Set`, `Map`, `HashSet`, `HashMap`, or `Lexicon`; understanding tradeoffs between various data structures
- **Implementing ADTs**: write code to implement operation(s) inside a basic ADT such as a list, stack, queue, etc., using standard implementation models (e.g., dynamic array, linked list, etc; depending on ADT)
- **Pointers/arrays**: write lines of code that use pointers and arrays, and/or look at a piece of pointer code and answer questions about it. Read and write code using operators *, `&`, `new`, and `delete`, as well as pointer arithmetic (adding/subtracting from pointers)
- **Linked lists**: write a few lines of code to change a "before" picture of some linked nodes into an "after" picture, and/or write functions that operate on a linked list
- **Algorithm analysis / Big-O**: look at a given piece of code and answer questions about its runtime complexity, and/or write a piece of code that solves a problem within a given Big-O limit
- **Recursion**: look at a piece of recursive code and write its output, and/or write a function that uses recursion to solve a problem
- **Backtracking**: write a function that uses recursive backtracking to solve a problem
- **Classes and objects**: write a class and/or add behavior/data to an existing C++ class; understanding public vs. private, `const`, constructors, operator overloading

Final exam topics study list:

- **Midterm topics**: see above.
- **Recursion and/or recursive backtracking in trees and graphs**: in addition to recursion as used on the midterm, read and write recursive code that operates on or explores trees and graphs
- **Trees**: simulate operations on a binary search tree (BST) such as adding or removing elements; perform various flavors of traversals on trees; tries as backing for Lexicon
- **Implementing data structures**: simulate operations on a data structure we implemented or discussed the implementation of in lecture, such as a map (BST or hashing), priority queue (heap or various from the assignment), stack, queue, vector; and/or, add behavior to an existing implementation

- **Searching**: trace the behavior of binary search, understand how searching is done in various data structures (array, tree, list, sorted and unsorted)
- **Sorting:** selection sort, insertion sort, merge sort, heap sort, quicksort (see practice exams for typical level of detail of study—note we did all of these in one day, and exam emphasis will reflect that)
- **Graphs**: write code to operate on graphs, which might include needing to know graph vocabulary such as connectedness, cycles, degree of vertices; simulate the execution of path-searching algorithms such as DFS, BFS, Dijkstra's algorithm, A*; simulating execution of Kruskal's algorithm; write code that operates on a graph (typically some basic traversal such as DFS/BFS, with customizations for the particular problem)
- **Inheritance, polymorphism, object-oriented programming**: look at a piece of code involving inheritance and interpret its output; and/or, write a class that uses inheritance to extend some existing base class

The following topics are guaranteed NOT to be required to solve any problem on the final exam:

- drawing fractals; graphics with the GWindow class
- multiple inheritance; private inheritance; initialization list in constructors in subclass
- overloading the = operator, copy constructors, and deep copying
- the Standard Template Library (STL)
- writing C++ template classes
- details of how to perform rotations on an AVL tree to restore its balance
- other kinds of advanced trees such as AVL, Red/Black, Splay
- anything else not explicitly covered in lecture or homework