C++ review material

- Very good knowledge of constructors, copy-constructors, and destructors).
- Very good knowledge of operator overloading (especially the assignment operator).
- Very good knowledge of pointers and dynamic array allocation.
- Differences between static and dynamic array allocation.

Chapter3 (Unsorted Lists, Sorted Lists, Big-O notation)

- Very good knowledge of the Unsorted List specification and implementation using arrays.
- Very good knowledge of the Sorted List specification and implementation using arrays.
- Implementation tradeoffs (e.g., in the case of unsorted lists, it is more efficient to insert at the end of the list).
- Binary search algorithm (be careful when "item" is not in the list).
- Comparison of algorithms and big-O notation **(very important)**
- Running times for all the functions of the Unsorted and Sorted List implementations.

Chapter4 (Stacks, Queues, and Templates)

- Very good knowledge of the Stack specification and implementation using arrays.
- Very good knowledge of the Queue specification and implementation using arrays.
- Implementation tradeoffs (e.g., using "front" and "rear" to determine whether the queue is empty or full).
- Running times for all the functions of the Stack and Queue implementations.
- Good understanding of the "template" mechanism.

Chapter5 (Linked Stacks, Linked Queues, Linked Lists)

- Very good knowledge of the Unsorted List specification and implementation using linked lists.
- Very good knowledge of the Sorted List specification and implementation using linked-lists.
- Very good knowledge of the Stack specification and implementation using linked-lists.
- Very good knowledge of the Queue specification and implementation using linked-lists.
- Running times for all the functions of the linked Stack, Queue, Unsorted List, and Sorted List implementations.
- Tradeoffs (e.g., memory or time) between array-based and linked-list-based implementations.

Chapter6 (Doubly Linked Lists, Inheritance)

- Insert/Delete in a doubly linked-list.
- Singly vs doubly linked-lists.
- Simplify coding using "Headers" and "Trailers".
- Implementing a linked-list using arrays.
- Inheritance (what it is, how it is implemented, static vs dynamic binding, virtual functions, slicing problem)

Chapter7 (Recursion)

- What is recursion and how to write a recursive function.
- Why are function calls expensive?

- Activation record and run-time stack.
- Iterative solutions vs recursive solutions (tradeoffs).
- Recursive implementation of binary search.
- Inserting items onto a sorted list recursively.

Chapter8 (Binary Search Trees)

- Definitions: binary tree, height, node level, ancestors, descendants, leaves.
- How to compute the height if we know the number of its nodes? Learn the proof.
- How to compute the number of nodes if we know its height? Learn the proof.
- Binary search tree property.
- Good knowledge of the Binary Search Tree specification and implementation.
- Inserting/Deleting nodes (understand them from an algorithmic point of view - don't memorize code) - Running times important.
- Tree traversals: inorder, preorder, postorder.
- Running times for all the functions of the Binary Search Tree type.

Chapter9 (Binary Expression Trees, Heaps, Graphs)

- Binary expression trees (purpose, advantages, evaluation, how to create them from a prefix or postfix expression).
- Full and complete trees (definitions).
- Array-based implementation of trees (why?).
- Very good understanding of the heap data structure (what it is and what kind of applications it is useful for).
- Reheap-up and Reheap-down (understand them from an algorithmic point of view - don't memorize code) - Running times important.

- What is a priority queue and what it is useful for.
- Good knowledge of the priority queue specification and implementation.
- Running time of Enqueue/Dequeue for priority queues.
- Other implementations of priority queues and tradeoffs (e.g., linked-list, binary search tree).
- Very good understanding of the graph data structure (what it is and what kind of applications it is useful for).
- Array-based vs linked-list-based implementation of graphs. Tradeoffs between the two implementations.
- Searching a graph using DFS or BFS (understand them from an algorithmic point of view - learn pseudo-code but don't memorize actual code)

Chapter10 (Sorting, Searching)

- Not included in the final exam.

Programming Assignments

There might be 1-2 questions based on the material of the programming assignments (not coding). The purpose is to test your knowledge on some of the ideas and algorithms we discussed in class for image manipulation.

General Comments

The final exam will be closed-books, closed-notes. The format will be the same as in the midterm (True/False, Short Answer Questions, Coding). Most of the questions (60%) will be based on the material we covered after the midterm.

• When studying the material from a particular chapter, study my notes first (there are quite a few things that are in the book but we did not cover in class and the opposite).

• Make sure you go over the examples we did in class that you can find in my notes.

• Do as many problems as you can from each chapter and review the questions of the midterm and of the quizzes (practicing is the key to do well in the final).

• Make sure you understand the *trade-offs* that exist between different implementations (e.g., array-based queues versus linked-list-based queues or array-based graphs versus linked-list-based graphs).

• Very important to know how to compute the running-time of an algorithm and how to express it in terms of big-O notation.