

Final Review

The final exam is cumulative. This review sheet complements the ones for tests 1 and 2.

- **Standard Template Library**

Basics: What is the STL? What are iterators and how are they used? How are STL classes used?

Basic containers (vectors, lists, queues, stacks): Main characteristics . How are they implemented? How efficient are various operations (e.g. inserting at the front vs. inserting at the back)? When would you use each one of them? Why don't stacks and queues have iterators? You should be able to create an empty container, add elements to it and traverse it.

Algorithms: You should be able to use the following algorithms: `find()`, `count()`, `sort()`, `remove()`, `remove_if()`, `find_if()`, `replace_if()`, `copy()`.

Function objects: Where do we use them? You should be able to create a function object and use it in conjunction with an algorithm.

- **Lists**

Implementation: What do we need to consider when implementing the list ADT using contiguous or singly/doubly linked memory?

- **Sorting Algorithms**

Basics: What are the main ideas behind insertion sort, mergesort and quicksort? What are their running times in the best/worst/average case? You should be able to apply them on an array of elements or choose the right algorithm for an application. What is a stable sorting algorithm?

- **I/O**

File I/O: What are streams? How is file I/O performed? What do the functions `peek()`, `seekg()`, `seekp()`, `eof()` do? How is unformatted I/O performed? How is `read()` different from `>>`?

- **Strings**

C++ strings: Hiding the memory book-keeping required by C-strings. Overloading `+`, `=`, `==`.

- **Exception Handling**

Traditional: What are the traditional methods for exception handling?

C++: What functionality does C++ provide? You should be able to write simple try-catch pairs, similar to the examples in the class notes.