Topics List for Exam 1

- Material from 3110
  - Classes:
    - Accessors and mutators (aka getters and setters
    - Private / public
    - What the =operator and copy constructors do by default and why this causes a problem with classes that have pointers in them.
    - Know how to implement constructors using member initialization lists.
    - Know when you would need a destructor.
    - What pass by value, pass by reference, and pass by const reference do, and why you would use/not use each one.
    - Operator overloading
      - Know how to overload the Boolean operators (==, !=, >, <, >=, <=)
      - Know how to implement the [] operator
      - Know how to implement the << operator
      - Know how to implement the + operator
  - Recursion
    - How to write a simple recursive algorithm
    - How to trace through a recursive algorithm
- Vectors
  - STL vectors
    - How to create a vector with an initial size
    - How to create an empty vector
    - How to initialize a vector using an array. (You'll need to recognize this, but you won't need to write it)
    - What the operations push_back(), pop_back(), back(), size(), empty(), and the [] operator do.
    - How to write functions that go through a vector, performing some operation (e.g. printing out a whole vector, summing up a vector, whatever).
  - Our own implementation
    - How did we implement a vector? (You don't have to implement one from scratch on the test, but you should know what we did)
    - What is the running time of push_back(), pop_back(), back(), size(), empty(), and the [] operator?
- Lists
  - STL lists
    - How to create a list with an initial size.
    - How to create an empty list.
    - How to initialize a list using an array. (You'll need to just recognize this.)

        - What the operations push_back(), pop_back(), back(), size(), empty(), push_front(), and pop_front() do.
        - How to write functions that go through an entire list using iterators, performing some operation.
    - Our Own list implementation
        - How did we implement a list?
        - Know how to write code to add/remove an element from the head/tail of a linked list structure.
        - What is the running time of push_back(), pop_back(), back(), size(), empty(), push_front(), and pop_front() using a linked list implementation.
- Analysis of Algorithms
    - Know the definitions for the following:
        - Running time (or time complexity)
        - Space complexity
        - Algorithm
        - Correctness of an algorithm
    - Know how to analyze the running time of a simple algorithm and explain why that is the correct running time.
    - Know how linear search and binary search work and which is better in which situations
- Stacks
    - Know how to create an empty stack
    - Know what push(), pop(), empty(), and size() do.
    - Be able to solve a problem using a stack.