Exam II next time!

I won't be having office hours today (CoE faculty meeting).

Reading from a file:

You have seen how to read from cin and write to cout. Reading and writing to a file is basically the same idea.
See LCR, ch. 5.

Objects of Vectors:
Look at the declarations of Card and Deck in skeleton form:

```
struct Deck; // forward declaration of Deck
struct Card {
    ….
    int find (const Deck& deck) const;
};

struct Deck {
    vector<Card> cards;
    …
};
```

The "forward declaration" breaks the cycle so things can be declared before they are used.

Sorting:

Searching and sorting are two of the most common things computers do. There are many many searching and sorting algorithms, and new ones being invented every day.

Selection sort is one of the simplest.
At any given point I am looking at one element of the vector. All the ones before it are already sorted.
So I swap that element with the smallest element in the remainder of the vector.

For an N-element vector I loop N times (in the outer loop).
In the inner loop I go N the first time, N-1 the second, N-2 the third and so on.
The number of times I go through the inner loop is approximately:
$N + (N-1) + (N-2) + \ldots + 2 + 1 = N(N+1)/2$.

So this is an $\mathcal{O}(N^2)$ or quadratic-time algorithm.

Can we do better? Merge sort is better for large N.

(1) We can merge two sorted decks in $\mathcal{O}(N)$ time (i.e., linear time time process, proportional to the size of the decks).

(2) I could take a big deck, split it into two or more smaller decks, sort the smaller decks, and then merge the results.

(3) If I take this idea to the limit, you keep splitting the decks in half until you get to a 1-card deck, which is already sorted.

If you analyze this algorithm (which takes some work), you find it takes O(N log N) time.

This is average case analysis. We can also do best case, worst case, or various conditions, such as the vector is already mostly sorted.

Object-oriented Programming:
We are going to be looking at classes of objects.
Structs and classes are identical except:
In structs, the members are by default public (but you declare them explicitly public or private).
In classes, the members are by default private (but you declare them explicitly public or private).