Day 22 (Mon 03/14)

- Day 22 recap questions
- Work on midterm project

Reminders:

- midterm project is due by 11pm on Friday
  March 18th
  - late submissions are not accepted!
  - submission on Gradescope coming soon
  - note: automated testing will be very limited

Day 22 recap questions:

1. What is the difference between C and C++?
2. What is a namespace in C++?
3. Why should you not put "using" statements in header files?
4. How do you read and write in C++ (i.e. standard inputting/outputting)?
5. What is the difference between C strings and C++ strings?
6. How long can a C++ string be?

1. What is the difference between C and C++?

C: "portable assembly language"

Very little direct support for rich data types.
Very little support for automatic resource management.
No support for generic programming.
Almost no run-time support is needed.
C is a relatively simple language.

C++: a modern object-oriented language

C++ classes allow the creation of very rich data types.
Constructors and destructors shift the burden of resource management
(memory, files, etc.) onto the compiler.
Extensive support for generic programming (template classes and functions.)
Some runtime support (e.g., exceptions) is required.
C++ is arguably the most complex programming language ever created.

2. What is a namespace in C++?

A namespace allows names of classes, functions, data types, etc. to be isolated so that they don't conflict with other classes/functions/data types that happen to have the same name.

E.g., all C++ standard library classes are in the std namespace. For example, std::string is the string class provided by the standard library. You could define your own class called "string", and it would not conflict with std::string.

If you are developing a library that you intend to be incorporated into other programs, put all of the classes/functions/data types/etc. in a namespace so that their names won't conflict with names used by the program and other libraries.

3. Why should you not put "using" statements in header files?

Because you would be forcing any code that #includes the header file to accept the name(s) imported by the using statement.

For example, if your header does

```
using std::string;
```

then you make it difficult for code #including your header to use any other class called "string".

4. How do you read and write in C++ (i.e. standard inputting/outputting)?

Writing data: use std::cout. Reading data: use std::cin.

Example:

```
std::string name;
int age;

std::cout << "What's your name? ";
std::cin >> name;

std::cout << "How old are you? ";
std::cin >> age;

std::cout << "Hello, " << name << ", nice to meet you\n";
```

5. What is the difference between C strings and C++ strings?

C strings:

Stored in an array of char.
Entirely the programmer's responsibility to ensure the string is not too large for the array.
Dealing with arbitrary-length strings (e.g., reading lines of text from a file) is quite tricky, and will generally involve dynamic allocation and resizing buffers on the fly.

C++ strings:

Storage for the character sequence handled automatically.
Storage grows as needed! And, is automatically freed when the string is destroyed.
Dealing with arbitrary-length strings is easy.

6. How long can a C++ string be?

As long as it needs to be (limited only by available memory.)