Using Dynamic Libraries

- You can load dynamically-linked libraries from within your program and call their functions by name
- On UNIX, `dlopen` opens a library by its filename
- `dlsym` looks up a function by name and returns a pointer to it
- Can call function through the pointer!
- Must know argument and return types
Why Your Own Containers?

• If you need to maintain collections of objects that are linked in more complex ways than in lists or similar structures
• If you want to control memory allocation
• If you want to control which operations are made most efficient, at the expense of others you rarely or never need
• If the structure of your collection may change, but you don’t want to have to change other code that uses it

Keeping Classes to Yourself

• If you don’t want just anyone using a class of yours, then make its constructors and destructor private
• To allow particular classes to use it, declare them as friend classes inside its own definition, so you control which classes have access!
• Form: friend classname;
• It’s all-or-nothing: friend classes can access any private members
Overloading ++, --

• To overload prefix ++, declare overload that takes no arguments if declared within a class, or one argument if not
• This overload must return a reference
• Make it increment, then return itself
• To overload postfix ++, add (unused) int argument, don’t return a reference
• Make it save itself, increment, then return original (saved) object

Overloading Dereferencing

• If a class serves as a wrapper class for an underlying object, you can overload * to support “dereferencing”, or extraction of the underlying object
• Implement one overload to return a const reference, which can’t be modified, and make function const too if it’s a member of a class
• Implement another overload to return a non-const reference
Next Time

There is no next time, we’re done!

Good luck finishing the project!

Happy Holidays!