

Weak Pointer Solutions

std::weak_ptr

- Briefly describe std::weak_ptr
 - An std::weak_ptr object can be created by copying an std::shared_ptr object
 - However, the weak_ptr does not affect the reference count
 - The weak_ptr will retain its pointer to the shared_ptr's allocated memory even after the memory has been released
 - std::weak_ptr has a very limited interface
 - The main use for a weak_ptr object is to convert it back to a shared_ptr
 - Before doing this, we must check that the weak_ptr object's memory pointer is still valid

weak_ptr to shared_ptr

- Give two ways in which a weak_ptr object can be converted to a shared_ptr object
 - Call the lock() member function
 - If the memory pointer is still valid, this will return a shared_ptr object
 - Pass the weak_ptr object to the constructor of shared_ptr
 - If the memory pointer is still valid, this will create a shared_ptr object
- What happens if the weak_ptr's memory pointer is invalid?
 - lock() returns nullptr
 - The constructor throws an exception bad_weak_ptr

weak_ptr to shared_ptr

- Write a program which creates a shared_ptr and a weak_ptr to it
- Convert the weak_ptr to a shared_ptr
 - When the original shared_ptr object is still valid
 - After the original shared_ptr object has been invalidated (this can be done by assigning it to nullptr)
- If it is safe to do so, print out the data in the resulting shared_ptr

Weak Pointer Applications

- Suggest an application for weak_ptr
 - Cache implementation
 - Use shared_ptr for the data
 - Use weak_ptr for the cache entries
 - Call lock() on the cache entry first
 - If successful, use the returned shared_ptr
 - If unsuccessful, fetch the data again