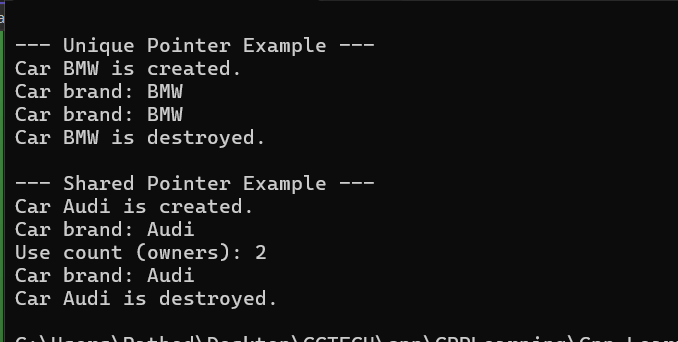
**Problem Statement:**

The problem is to manage dynamic memory in a safe and efficient way, avoiding memory leaks, dangling pointers, and other issues that arise when using raw pointers in C++. We will demonstrate the use of std::unique\_ptr, std::shared\_ptr, and std::weak\_ptr smart pointers to solve this.



**Learnings-**

 **Unique Pointer Example**:

* We create a Car object using unique\_ptr, which ensures exclusive ownership.
* When the ownership is transferred (via std::move), the previous owner loses access to the object.

 **Shared Pointer Example**:

* We create a Car object using shared\_ptr. Multiple shared\_ptr objects can manage the same memory. We track the reference count using use\_count() to show how many owners there are.