Smart pointers are a specific kind of "scoping object". They are like regular pointers but can automatically deallocate the object they point to when they go out of scope. Since C++ is not a garbage collected language, such functionality is important. The pattern where scoping objects are used to automatically manage the lifetime of heap-allocated objects is called RAII - Resource Acquisition Is Initialization.

Here's some sample use of scoped\_ptr, the most common type of smart pointer:

// We put a pointer into a smart pointer at construction time.

scoped\_ptr<Foo> foo\_ptr(new Foo(...));

// ...Or by using reset().

scoped\_ptr<Bar> bar\_ptr; // Like "Bar\* bar\_ptr = NULL;".

bar\_ptr.reset(new Bar(...)); // Now |bar\_ptr| is non-NULL and owns the object.

// We can test the smart pointer by using get() to see the raw pointer underneath.

if (!value.get())

return false;

Foo\* raw\_ptr = foo\_ptr.get();

// We can call through the smart pointer as if it were a pointer.

scoped\_ptr<Foo> foo\_ptr(new Foo(...));

foo\_ptr->foo(...);

// ... Or by \*

(\*foo\_ptr).foo(...);

Now let’s implement this class. You need to implement the following member functions.

1. Constructor and destructor
2. reset: reset current pointer. Note: You need to delete the previous one. However, sometimes you didn’t need to delete it.
3. get: return the raw pointer.
4. Operator Overloading for "\*" and "->"

Hint: Template, constructor and destructor, new and delete, Operator overloading. You should submit the definition and implementation of the scoped\_ptr class.