

Writing a Class using RAII Exercises

- Explain how using RAII ensures the strong exception guarantee for
 - Constructor
 - If an exception is thrown when acquiring the resource (e.g. calling new), the constructor returns immediately
 - The partially-constructed object and its members are destroyed
 - Copy constructor
 - Same as for constructor

- Explain how using RAI ensures the strong exception guarantee for
 - Assignment Operator
 - The assignment operator which swaps a temporary provides the strong exception guarantee by using:
 - Copy constructor which offers the strong exception guarantee
 - Swap operation which does not throw
 - Destructor which does not throw
 - Destructor
 - Implemented using noexcept operations

- Explain why the assignment operator, as usually written for an RAI class, offers only the basic exception guarantee
 - This assignment operator manually acquires a new resource (e.g. calls new) after releasing the old one (e.g. calling delete)
 - If the acquisition step throws, the old resource cannot be recovered
 - The object is not in its original state, so we do not have the strong guarantee
 - However, no resources have been leaked, so we have the basic guarantee

- Explain in detail how the function shown below works
 - A temporary copy is made of the “other” instance
 - This will invoke the copy constructor
 - If the copy constructor throws, the temporary instance will be destroyed and the assignment operation will return
 - The “this” instance is unaltered

- Explain in detail how the function shown below works
 - The “this” object is then swapped with the temporary copy
 - This swap operation does not throw
 - After the swap, “this” will have the same value as the temporary copy, which has the same value as the “other” instance
 - i.e. “this” has been successfully assigned to
 - Finally, “this” is returned by reference, which cannot throw
 - The temporary copy, which now contains the old value of “this” is destroyed

- Explain why this offers the strong exception guarantee
 - The only stage at which an exception could be thrown is the copy
 - If an exception is thrown here, the assignment operator will return with “this” in its original state
 - Hence this assignment operator provides the strong exception guarantee

- Describe some of the other advantages this implementation has over the version without a temporary
 - Concise code which is easier to get right
 - Works correctly even with self-assignment
 - Re-uses code (from copy constructor and destructor) instead of duplicating it

- Implement the "rule of three" operators for the class shown below using the RAII idiom

```
class BufferManager {  
    private:  
        int size;  
        char *buffer;  
        ....  
};
```