Move Operators Solutions

Move operators

- What are the names of the move operators?
  - Move constructor and move assignment operator
- Briefly explain their purpose
  - Move constructor initializes an object from an rvalue by moving the argument's data into the new object
  - Move assignment operator assigns an object from an rvalue by moving the argument's data into the target object
- Write down prototypes for the move operators for a class Test

```
Test(Test&& other) noexcept;                          // Move constructor
Test& operator =(Test&& other) noexcept;              // Move assignment operator
```

Move operator syntax

- What differences in syntax are there between move operators and their copy equivalents?
- Briefly explain why these differences exist
  - Move operators take their argument by rvalue reference instead of const lvalue reference
  - This is because the argument will be moved from
  - Move operators should be declared noexcept
  - This is because recovering from a half-completed operation is much harder for a move than it is for a copy
  - Also because STL containers will not call an element's move operator unless it is noexcept

Move Operator Implementation

- Add copy and move operators to the following class
- Each operator should print out a message when it is called, to say what kind of operator it is
- Write a program to test your class
- Explain your results

```
class Test {
  private:
    int i{0};             // Built-in member
    MyClass m;            // Class member
  public:
    Test() = default;     // Constructor (calls m's constructor)
};
```

Move Operator Implementation

- Explain your results
  - The copy constructor is called when initializing a new object from a variable
  - The move constructor is called when initializing a new object from an rvalue
  - The copy assignment operator is called when assigning an object from a variable
  - The move assignment operator is called when assigning an object from an rvalue
  - When initializing a new object from a temporary object, you may not see any output
  - This is because the compiler is allowed to use an elided call to the copy constructor and create the new object directly, if it is more efficient than creating a temporary and calling the move constructor.

Move operators for derived classes

- When writing a move operator for a derived class, are there any special considerations you should bear in mind?
  - The derived class's operator should call the base class's operator, so that the whole object is moved and not just the derived part of it
  - To ensure the move version of the base class's operator is called, its argument should be cast to an rvalue

```
// Call Base class's move constructor from Derived class's move constructor
Derived(Derived&& other) : Base(std::move(other)) noexcept { ... }
```

Compatibility with older versions of C++

- What happens if we have a class that was written for an older version of C++ and does not have any move operators?
  - The class will behave in the same way as it did before
  - An rvalue object of this class will be copied, or copy assigned (assuming it has copy operators)