Copy Elision Solutions

Copy Elision

- Explain briefly what is meant by copy elision
  - Copy elision occurs when the compiler omits a call to the copy constructor
  - Usually this is done when initializing a new object from a temporary object
  - Instead of creating a temporary object, initializing the new object from the temporary and destroying the temporary, the compiler creates the new object directly with the same arguments that would have been given to the temporary object

Function returning temporary object

- Write a simple class. Implement the copy constructor to display a message when called
- Write a function which returns a temporary object of the class by value
- Write a program which calls this function and uses its result to initialize an object of the class

Function returning temporary object

- How many times will the copy constructor be called?
- Run the program. Explain your observations
  - Without optimization, the copy constructor would be called twice, once for the return by value and once for the object initialization
  - Modern compilers will probably elide both calls, so the copy constructor is not called at all

Return Value Optimization

- Explain what is meant by "Return Value Optimization" and "Named Return Value Optimization"
  - In return value optimization, a temporary variable is returned by value and the compiler elides the copy constructor call
  - In named return value optimization, a local variable is returned by value and the compiler elides the copy constructor call
- Write a simple program which demonstrates Named Return Value Optimization

Copy Elision in C++17

- How have the copy elision rules changed in C++17?
  - Before C++17, copy elision was optional
  - In C++17, the compiler must elide copies when initializing from temporary objects, or when returning a temporary object by value (RVO)
  - Copy elision is still optional when returning named objects by value (NVRO)
  - In effect, these rules confirm existing practice