

Copy constructor elision Exercises

- What is meant by "copy constructor elision"?
 - The compiler is allowed to omit copy constructor calls to optimize the code, even if it changes the behaviour of the program
- Give some examples where copy constructor elision occurs
 - Inserting temporary instance into container
 - Returning temporary instance by value

- What is meant by "return value optimization"?
 - When returning a temporary instance by value, the temporary instance is created directly in the function's return value space instead of being copied into it
- What is meant by "named return value optimization"?
 - When returning a local variable by value, the temporary instance is created directly in the function's return value space instead of being copied into it

- If the code on the next page is executed, how many times will "Copy made" be printed out?
 - Between 0 and 2 times
- Explain your answer
 - The program would normally make two copies of an X instance
 - Once when copying the local variable x into func's return space
 - Once when copying the returned value from func into main's variable x
 - However, the compiler may elided either or both of these copies

- Convert the code to a working program and run it. What do you observe?
 - On my system, there was no output, indicating that both the copy constructor calls were elided
- If your compiler supports `-fno-elide-constructors`, compile the program with this option and run it again
 - On my system, “Copy made” appeared twice, indicating that both the copy constructor calls were executed

```
class X {  
    public:  
        X() {}  
        X(const X& other) { cout << "Copy made\n"; }  
};
```

```
X func() {  
    X x;  
    return x;  
}
```

```
X x = func();
```