Modern C++ Overview
Part Four Solutions

Rvalue references and overloading

- Write a function which takes an int by const reference and an overloaded version which takes an int by rvalue reference
- The function displays its argument type
- Write a program which calls the overloaded function and passes
  - An int variable
  - The result of calling std::move() on an int variable
  - An integer literal

Rvalue references and overloading

- Explain your results
  - An int variable is an lvalue (we can take its address), so the lvalue overload is called
  - std::move() casts its argument to an rvalue, so the rvalue overload is called
  - A literal is an rvalue (we cannot take its address), so the rvalue overload is called

Move-only types

- What is meant by a move-only type? Give an example of a move-only type
  - A move-only type cannot be copied, but it can be moved
  - Examples: fstream, unique_ptr
- Why are move-only types useful?
  - Move-only types are useful when a class owns a resource and manages its lifetime. In order to give an object sole ownership of the resource, copying cannot be allowed. However, the resource can be transferred from one object to another, in which case the target object becomes the new owner of the resource.

Pass by move

- What property must a class have in order that objects of that class can be passed by move?
  - The class must define a move constructor

Move operators

- Write down the prototypes of the move constructor and move assignment operator of a class called "myclass"

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