Modern C++ Overview
Part Five Solutions

Lambda-local Variables

- How can we create a variable which is local to a lambda body?
  - Put the variable, with an initializer, in the capture block
- Does the variable need to be declared auto?
  - No, the type is auto by implication
- Does the variable need to be initialized?
  - Yes, to allow the compiler to deduce its type
- Write a program with a lambda expression which has a variable that is local to its body

Lambda-local Variables

- Can we initialize a lambda-local variable from a captured variable?
  - Yes, just use the captured variable as the initializer
- What syntax do we use to capture the variable?
  - None. The variable is captured by implication
- Write a program which uses a lambda with a local variable which is initialized from a captured variable

Lambda-local Variables and Capture by Move

- How can a lambda expression capture by move?
    - Use a lambda-local variable which is initialized from an rvalue
- Write a program in which a lambda expression captures by move

Random Number Classes

- Write a program which prints out 10 random integers between 0 and 100
- Write a program which prints out 10 random floating-point numbers between 0 and 1

Random Number Classes

- Why is it generally a bad idea to use a local variable for a random number engine?
  - Creating an engine is fairly time-consuming
  - Creating a new engine will reset the sequence
  - Usually you will only need one instance per program anyway

std::unique_ptr

- Briefly describe how std::unique_ptr is implemented
  - unique_ptr is a class which has a traditional pointer as a private data member
  - It has public member functions which implement some of the features of traditional pointers
- In terms of memory usage and efficiency, how does unique_ptr compare to traditional pointers?
  - There is no extra overhead from using a unique_ptr instead of a traditional pointer

std::unique_ptr and RAII

- Explain how unique_ptr follows the principles of RAII
  - The allocated memory is managed by the class
  - It is acquired or allocated in the class's constructor
  - It is released in the class's destructor
  - The class manages transfer of ownership of the traditional pointer from one object to another

unique_ptr initialization

- Write a simple program that creates and initializes a unique_ptr object and performs some operations on it
- What changes would you need to make your program compile under C++11?
- (Optional) Put your compiler into C++11 mode and check your answer to the previous question