HW 14

Turn in a .tar or .zip file containing the program for 2. DO NOT turn in any binaries created.

1. Determine what is printed by each call in HW14a.cpp.
2. Write a C++ program with at least two classes, *Mammal*, *Canine* and *Pet* class. Unlike the program in HW14a.cpp you should use separate .h and .cpp files.
   a. The *Mammal* class should support the following functionality
      i. A private *legs* field that specifies the number of legs and that can only be initialized/set once.
      ii. A public getter function for *legs* that cannot be overridden. See http://stackoverflow.com/questions/4465686/how-to-prevent-a-method-from-being-overriden-in-derived-class/16896559#16896559
      iii. A constructor that initializes the *legs* field and that takes an *int* argument
   b. The *Canine* class that should inherit *Mammal* and implements the following functionality:
      i. A constructor that initializes the *legs* field to *4* via a call to the *Mammal* class constructor and that initializes all fields in the the *Canine* class. The constructor should take as many arguments as necessary.
      ii. Commented out attempt to override the getter function.
      iii. A virtual function that returns the breed of the dog (and associated private field(s), if necessary). You can make up breeds or find real ones.
   c. A *Pet* class that inherits from *Canine* and implements the following functionality:
      i. A function that overrides the breed function in the *Canine* class and adds "AKC: " to the breed name before returning it.
      ii. A constructor the properly causes all of base classes to be properly initialized.
   d. A main function that creates a *Pet* object, demonstrates how to call all functions visible in *Pet* using polymorphism, if they are virtual functions. It will also show how all functions visible in *Pet* can be called non-polymorphically, even if *virtual*.