Sean Armbruster   
Ch. 15 Review Questions   
pp. 977-297: 1, 2, 4, 5, 6, 7, 8, 10, 21, 22, 23, 24, 25, 26, 27, 28, 32, 34, 37

1. An “is a” relationship is when one object is a specialized version of another object.
2. Regarding the two classes – Dog and Poodle, Dog would be the base class and Poodle would be the derived class.
3. Protected class members act like private members, but they may be accessed by derived classes whereas Private members can only be accessed in the class in which they are declared, thus excluding derived classes.
4. A derived class cannot directly access Private members of its base class.
5. The base class’s constructor is called before the derived class’s constructor.
6. An overloaded function is one with the same name as more or other functions, but with different parameter lists to let the compiler know which one is which when called. A redefined function happens when a derived class has a function with the same name and parameters as the base class. The name and parameters may be the same because the derived class function is always called by objects of the derived class type.
7. Static binding happens at compile-time while dynamic binding happens at runtime.

10. No. The Potato and Vegetable Class are derived from one base class – Food. This situation may be called a “chain of inheritance.”

21. Last  
22. Scope Resolution Operator   
23. Base Class Version   
24. Virtual   
25. Static   
26. Dynamic   
27. Polymorphism   
28. Overriding / Virtual   
32. Ultiple Inheritance   
  
34.   
 class Poodle : public Dog   
37.  
  
class B   
{  
private:   
 int m;  
protected:   
 int n;   
public:  
 void setM( int M) { m = M; }  
 int getM() { return m; }  
 void setN( int N ) { n = N; }  
 int getN() { return n; }  
 virtual int calc() { return n\*m; }   
};  
  
class D : public B   
{  
protected:  
 float q;  
 float r;  
public:   
 void setQ(float Q) { q = Q; }  
 float getQ() { return q; }   
 void setR(float R) { r = R; }  
 float getR() { return r; }  
 virtual int calc() override { return r \* q; }  
};