1. Upcasting is when you are casting up to a superclass, while downcasting is casting down to a subclass. If country is a base class, and china is a subclass, then casting china to country would be considered an upcast; and we are able to do this because china is a country. Downcasting works when you know at runtime, that it is legal to cast, because downcasting only checks at compile time.

2.You would mark a method as virtual when you want to override the method’s superclass. You would also mark a method as virtual when creating abstract methods. You would not want to mark a method as virtual if it is not overriding the method’s superclass or making an abstract method because it will unnecessarily slow down your program.

3. Dynamic casts look at the object being cast, and checks to see if it is legal to do, returns a null pointer if it is illegal. Object that is being dynamic casted on needs to be polymorphic, meaning that the base class needs to inherit form the superclass. Dynamic casts check object and runtime.

Static casts: Checks object at compile time, not runtime. Static casts will still compile even if the superclass contains an object other than the base class that is using static cast to downcast.

You would use static casts when you know for sure that when casting it is legal at runtime, and you want a little more speed.

When speed is not a concern, using dynamic casts is safer because it checks the cast at runtime, and will not get unexpected results if the cast at runtime isn’t “correct”, instead it will return a null pointer.

4. Const casts is to get rid of all consts in an object/class. This is a good idea when using another library, or code you cannot modify. Using a const cast is ideal when the author of the code or library misused const (either omitting it, or adding it where it shouldn’t be).

5. Void eat(Fruit f), the method can modify the banana object, also this is pass by value and it makes a copy of banana, which anything modified will not be saved to the banana instance that was passed. while the void eat(const Fruit &f) passes the banana as a reference, and the banana cannot modify the fruit class.

6. In java, every class can only have one parent class, in c++, a class can have multiple parents, or no parents. In java, all classes inherit the object class as the base class. Java’s way of dealing with multiple inheritance in a safe way was to use interfaces. Interfaces just have the method name, and parameters of the method. An interface class could be thought of as marking all methods as virtual in the superclass, with empty bodies. The issue with multiple inheritance with c++ is that if methods have the same arguments and name in both super classes that your inheriting from, the question then becomes which method will be used.

7. I have provided 2 arguments in main but for some reason, it says i did not provided 0 arguments when compiling.