Name: Tai Bui
CSCI 3731 – Hw09

1. What is the difference between “downcasting” and “upcasting”?

While “upcasting” is to refer to an object as though it was its super class, “downcasting” is to convert an object back to a subclass.

2. Why would you mark a method as virtual? Why wouldn’t you?

Marking a method virtual mean that a subclass can override that method which is defined in the superclass. Without the virtual keyword, the program would call the method of the variable type, not the method of the actual class.

3. What is the difference between a dynamic cast and a static cast? Why would you use one over the other?
Dynamic cast works just like Java casting, it looks at the actual type of the object being casted and checks if it is legal to cast it to the requested type. On the other hand, static cast checks what it can at compile time, but do not look at the actual type of the object at runtime. Though static cast is faster, it is not as safe as dynamic cast for casting object. If super class has two different subclasses, while dynamic cast may fail to cast, static cast will force casting even though subobject1 is casted as subobject2.

4. When might you use a const cast?
When calling some library in which the author forgot to mark some methods argument as const even though the method does not modify the argument. Thus, const cast allows users to be able to pass the return of these non-const methods into const objects.

5. Suppose you have a Fruit base class and a Banana subclass. Suppose you have a banana declared like this:

Banana b;
What is the difference between what happens when you pass your banana to a method declared like this:
void eat(Fruit f);
vs. one declared like this:
void eat(const Fruit& f);.
When passing to argument Fruit f, the function gets a copy of the Fruit object, not a Banana object. Thus, it will execute the eat method in the Fruit class, not the Banana’s eat method.

When passing to argument const Fruit& f, the function takes a reference to a const Fruit object. Thus, it will execute the eat method in the Banana’s class.

6. What is the difference between how Java and C++ handle multiple inheritance?

In Java, every class has one and only one parent class. To overcome such restriction, Java allows the use of interface to group together classes in ways that violate the normal inheritance tree. In C++, a class can have no parent or multiple parents, each with its own data and method implementations. The problem arises when both super classes have same methods or inherit from same parent. Though this is a soluble problem, it is difficult and complicated to tackle such compared to the use of interface in Java. It is recommended to let one of the superclasses have no data and only have pure virtual methods.