Lab 13 – Multiple Inheritance

Estimated time: 15 minutes

In this lab you will complete the implementation of the following abridged class hierarchy:

```
class FlyingAnimal
{};

class ZooAnimal
{};

class Eagle: public ZooAnimal, public FlyingAnimal
{};
```

Begin with code provided in the starter folder. Declare variables of each type, both with and without arguments. Run the program and make sure you understand the sequence of constructor and destructor calls.

Notes:

1. Use the member initialization list to invoke the constructors for ZooAnimal and FlyingAnimal from the Eagle constructors.
2. Have each constructor and destructor print out a message so you can tell which are called.

Add the following class to the animal hierarchy:

```
class Animal
{
  public:
    int m_weight;

    Animal(int weight);
    Animal();
   ~Animal();
```

Derive both ZooAnimal and FlyingAnimal from Animal.

Notes:

1. Adding the new class at the top of the hierarchy will require changes to all the derived class. Specifically, the constructors with arguments will now need an extra parameter for the m_weight data member.
2. The Eagle class will now receive two copies of the base class Animal. Try setting both m_weight members directly using the scope resolution operator to eliminate the ambiguity.

Of course it does not really make sense for any animal to have two weights. Make the inheritance relationship between Animal and its derived classes virtual.

Notes:

1. Note that now the default constructor for Animal is called when constructing an Eagle, even when passing arguments to the ZooAnimal and FlyingAnimal constructors. Under virtual inheritance, to pass arguments to the virtual base class Animal, you must explicitly mention it in Eagle's member initialization list.

```
Eagle::Eagle(int weight, int minCageArea, int
   wingSpan) :Animal(weight),
   ...
   {
   }
```