CS 3721: Programming Languages Lab
Lab #13: Memory layout of class objects.

Suppose we have the following C++ code.

```
class Animal {
   protected:
     float age, weight;
   public:
     Animal(float _age, float _weight)
       : age(_age), weight(_weight) {} // initialize private variables
     void jump() { ... }
     virtual void grow(float time, float w) { age += time; weight += w; }
};

class Dog : public Animal {
   private:
     int tagno;
   public:
     Dog(float _age, float _weight, int _tag)
      : Animal(_age,_weight),tagno(_tag){} // initialize base and private
     virtual void feed(int num) { ... }
     virtual float grow(float w)  { ... }
};

class Dolphin: public Animal {
  private:
    float size;
  public:
    Dolphin(float _age, float _weight, float _size)
     : Animal(_age, _weight),  size(_size) {} // initialize base and private
    virtual void jump()  { ...}
};

void Zoo()
{
  Dog dog1(1.1,5,1);  // allocating a Dog object on stack
  dog1.feed(3);
  Dolphin *dolphin2 = new Dolphin(2.5, 7.9, 20);
  Animal *p1 = &dog1, *p2 = dolphin2;
  p1->jump();
  p2->jump();
}
```

The following describes the memory layout for the three classes Animal, Dog, and Dolphin.

```
Animal : vptr -> vtable: grow_float_float -> impl of Animal::grow
         age  : ?
         weight : ?

Dog : vptr -> vtable: grow_float_float -> impl of Animal::grow
```

```
                    feed -> impl of Dog::feed
                    grow_float -> impl of Dog::grow_float
      age  : ?
      weight : ?
      tagno : ?

Dolphin: vptr -> vtable: grow_float_float -> impl of Animal::grow
                         jump -> impl of Dolphin::jump
         age  : ?
         weight : ?
         size : ?
```

Further, the following draws the layout of variables allocated by the function Zoo.

```
     dog1 : vptr ->   vtable: grow_float_float -> impl of Animal::grow
                     feed -> impl of Dog::feed
                      grow_float -> impl of Dog::grow_float
            age  : 1.1
            weight : 5
            tagno : 1
     dolphin2 ->  vptr -> vtable: grow_float_float -> impl of Animal::grow
                                  jump -> impl of Dolphin::jump
                  age  : 2.5
                  weight : 7.9
                  size : 20
     p1  -> Zoo:dog1
     p2 -> StorageOf(Zoo:dolphin2)
```

1. Suppose we have the following C++ code.

```
class Plant {
     int location;
   protected:
     float age, height;
     int group;
   public:
     Plant(float _age, float _height, int _group)
       : age(_age), height(_height), group(_group), location(0) {}
     virtual void move(int _loc) { location = _loc; }
     void grow(float time, float h) { age += time; height += h; }
};

class Rose : public Plant {
   private:
     int flowers;
   public:
     Rose(float _age, float _height, int _group, int _flowers)
      : Plant(_age,_height,_group),flowers(_flowers){}
     virtual void water(int amount) { ... }
     virtual void move(int _loc)  { ... }
};
```

```
class Oak: public Plant {
  private:
    float density;
  public:
    Oak(float _age, float _height, int _group, float _density)
     : Plant(_age, _height, group),  density(_density) {}
    virtual void grow(float t, float h)  { ...}
};

int main()
{
  Rose rose1(1.1, 5.0, 1, 5);  // allocating a Rose object on stack
  Oak *oak2 = new Oak(2.5, 7.9, 20, 9.1);
  Plant *p1 = &rose1, *p2 = oak2;
}
```

(a) Give the memory layout for the three classes Plant, Rose, and Oak.

(b) Give the layout of variabless allocated by the function *main*.