1. Implement MergeSort in C++, accepting an array of integers as input. Exchange solutions with your classmates (to learn who wrote the best code)

2. Consider the following class declarations in C++:

```
class C {
   protected: int x;
   public: void f(){...};
};

class C1: public C {
   protected: int x1;
   public: void h(C *obj){...};
};

class C2: public C1 {
   public: int x2;
};

class C3: public C {
   public: f(){...};
};
```

a. Draw the class hierarchy
b. Assume that main contains the following declaration:

   C1 obj1;

   For each of the following expressions, say whether it is allowed in the code of main or not (they can be forbidden either because they violate the class definition or the protection mechanism)

   obj1.x , obj1.f() , obj1.x1 , obj1.x2

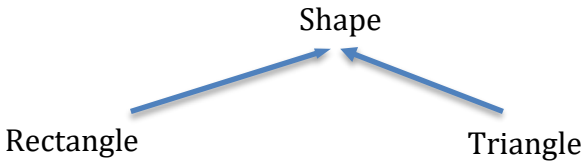
c. Assume that the body of C1::h contains the following declarations

```
C2 *obj2;
C3 *obj3;
```

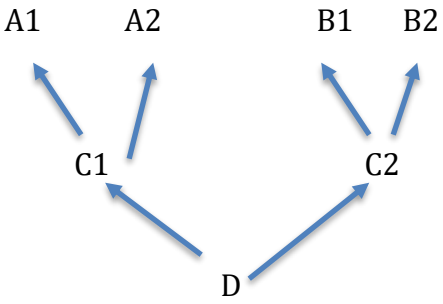
   For each of the following expressions, say whether it is allowed in the body of C1::h or not

   obj->x , obj2->x , obj3->x

3. Refer to the following figure. Write a C++ program, defining base class and derived classes as well as instantiating objects as in the figure.



4. Define classes according to the following inheritance hierarchy. Explore the order of execution of constructors when an object of class D is created.



5. Create a simple "shape" hierarchy: a base class called **Shape** and derived classes called **Circle, Square,** and **Triangle.** In the base class, make a virtual function called **draw( )**, and override this in the derived classes. Make an array of pointers to **Shape** objects while each element is either a **Circle,** a **Square,** or a **Triangle** (and thus perform upcasting of the pointers), then call **draw()** through the base-class pointers, to verify the behavior of the virtual function. (This is similar to the example presented in the lecture note.)