Part 1:

The display shown for find *LeastAndMostExpensiveUAV()* is **correct.**

How function works:
Let's walk through the function. First a for loop is used to see how many objects in the passed array are a UAV, and the number is stored in a variable. If the counter stays at zero (meaning no UAV objects were found), then the function is finished and the user is told that no UAV objects have been found in the array. If however any UAV objects are found, then an array of UAV objects is created with size equivalent to the number of UAV objects found in the original passed array. Then the original array passed into the function is looped through, and each UAV object is put into the new UAV array. The first UAV in the UAV array is set to be the most expensive and the least expensive, then a for loop is used to loop through the UAV array and if an object is more of less expensive than the current most or least expensive UAV, then it becomes the most or least expensive UAV (the index of the least and most expensive UAV in the UAV array is stored in a respective variable). Then the least and most expensive UAVs are printed out for the user to see.

Why the function works:
The secret to this function is the instanceOf operator. Without the instanceOf operator, it is difficult and much less elegant to determine if the object is a UAV. Once the UAV array is created and filled with UAV objects, keeping track of the index's of the current most and least expensive UAV objects in the array allows for easy comparison between the UAV object's prices.

Part 2:

The display shown for *copyFlyingObjects()* is **incorrect.**

How function works:
First a new array of FlyingObjects with size equal to the one passed in the array is created. Then for each object in the array passed to the function, a new FlyingObjects object is created with the copy constructor of FlyingObjects and stored in the new array. Then the new array is returned.

Why the function does not work:
The problem with this function is that the class of each object is not found and taken into account. So if an object of class Airplane is passed into the FlyingObjects copy constructor, a new object will be created (which is what we want), but this object will lose all of its attributes specific to the Airplane class. Since the FlyingObjects class is only used to group all of our other classes together into one hierarchy, it only has one variable which is objectNumber. So once all the objects in our original array are copied and stored into the new array, they all lose their attributes and are left with only an object number. From the object number we can see that they are in fact new copies of objects, not just pointing to the same object, but since all of the attributes are lost the function is useless.