1. In the Java Collections Framework, `List` is an interface that is implemented by `ArrayList`. For each of the statements below, indicate if it is a valid statement with no compilation error. Explain why.

   (a) Problem #A

   ```
   1  void foo(List<?> list) { }
   2  // code omitted
   3  foo(new ArrayList<String>());
   ```

   (b) Problem #B

   ```
   1  void foo(List<Object> list) { }
   2  // code omitted
   3  foo(new ArrayList<String>());
   ```

   (c) Problem #C

   ```
   1  void foo(List<? super Integer> list) { }
   2  // code omitted
   3  foo(new List<Obejct>());
   ```

   (d) Problem #D

   ```
   1  void foo(List<? extends Object> list) { }
   2  // code omitted
   3  foo(new ArrayList<Object>());
   ```

   (e) Problem #D

   ```
   1  void foo(List<? super Integer> list) { }
   2  // code omitted
   3  foo(new ArrayList());
   ```

2. The following static generic method `max3` that takes in an array of generic type `T` such that `T` implements the `Comparable` interface.

   ```
   1  static <T extends Comparable<T>> T max3(T[] arr) {
   2    T max = arr[0];
   3    if (arr[1].compareTo(max) > 0) {
   4      max = arr[1];
   5    }
   6    if (arr[2].compareTo(max) > 0) {
   7      max = arr[2];
   8    }
   9    return max;
   10 }
   ```

   What happens if we replace the method header with each of the following:

   (a) `static <T> Comparable<T> max3(Comparable<T>[] arr)`

   (b) `static <T> T max3(Comparable<T>[] arr)`

   (c) `static Comparable max3(Comparable[] arr)`

3. Suppose a `Fruit` class implements `Comparable` interface, and an `Orange` is a subclass of `Fruit`. How would you change the `max3` method header in Question 2 such that the parameter type of `max3` is `List<T>` instead? You should aim to make the method as flexible as you can.