Solution to CSE143 Section #2 Problems

1. One possible solution appears below.

        public boolean isPairwiseSorted() {
            for (int i = 0; i < size - 1; i += 2) {
                if (elementData[i] > elementData[i + 1]) {
                    return false;
                }
            }
            return true;
        }

2. One possible solution appears below.

        public void mirror() {
            int last = 2 * size - 1;
            for (int i = 0; i < size; i++) {
                elementData[last - i] = elementData[i];
            }
            size *= 2;
        }

One possible solution appears below.

        public ArrayIntList fromCounts() {
            ArrayIntList result = new ArrayIntList();
            int size2 = 0;
            for (int i = 0; i < size; i += 2) {
                for (int j = 0; j < elementData[i]; j++) {
                    result.elementData[size2] = elementData[i + 1];
                    size2++;
                }
            }
            result.size = size2;
            return result;
        }

4. Below is a list of style problems with the bad ArrayIntList:

        class comment: Don't include Stuart's name, include just your name. It
        would also be helpful to include the date and your section or TA's name.
        The class comment is meaningless. Make some kind of attempt to describe
        what the class is used for, as in "Class ArrayIntList can be used to store
        a list of integers."

        fields: Fields should be declared private. The field comments are useless
        because they are repeating the names of the fields. Only include comments
        if you can provide something beyond the field name. The field called
        "capacity" is not needed because it has the same value as
        elementData.length.

        class constant: It is improperly declared because it is missing "final".
        It is also improperly named because the convention for constants is to use
        all uppercase letters and underscore characters, as in DEFAULT_CAPACITY.

        first constructor: It should have a comment and it should be using the
        "this(...)" notation to call the other constructor. It does not use the
        class constant as it should.

second constructor: Comment doesn't mention the fact that it throws an
IllegalArgumentException when capacity is negative and the comment should
describe more about what it does. The use of if/else is not appropriate.
The convention in Java is to throw exceptions with an if and then to have
the standard code follow without being inside an else.

size method: It has no comment.

get method: Comment doesn't mention what kind of exception is thrown and
doesn't describe what it does. The for loop is not needed and makes the
method extremely inefficient, basically negating the benefit of using an
array (the random-access aspect of the array).

toString method: No comment. Spacing is terrible. Introduce spaces to
make your code more readable. The indentation is also off on many lines.

contains method: The comment has implementation details, discussing the use
of a for loop and the fact that it is searching an array. This method is
also highly redundant. It should call indexOf. The if/else after the loop
violates boolean zen (can simply return (count > 0)).

indexOf method: The comment has implementation details, talking about the
array and the size fields. It also does not describe significant behavior:
the fact that the first occurrence is returned and that a -1 is returned if
not found. The implementation is horrible. It uses a variable called
index that is only needed because the loop goes backwards and then there is
a redundant test after the loop. Even if you are going to use this index
variable, then initialize it to -1 and return it after the loop instead of
having the same test both inside and outside the loop (this fixes a minor
bug where it can return an index of 0 when the list is empty).

first add method: This is a good method.

second add method: The exception comments are good, but the description of
what it does is incomplete. It doesn't say what happens to the old value
at the given index. It shifts subsequent values to the right, which should
be described in the comment so that the client knows what it does. Also,
the loop structure is very bad. Code that is executed once either before
or after the repeated operation of a loop should appear outside the loop.
In particular, The exception checking should occur before the loop and the
"else" part that stores the value and increments size should appear after
the loop. Because the exception testing is inside the loop, it never
throws an exception for index values greater than the size of the list.

capacity method: You are not allowed to add extra public functions to a
class that weren't part of the specification. You can add private methods
that are part of the implementation, but not public methods.

remove method: The comment shouldn't discuss the implementation detail of
decreasing the size and it should mention what happens to the list when the
value is removed. The subsequent values are shifted to the left.

addAll method: The comment does not mention the parameter (i.e., that the
values being appended come from the other ArrayIntList). It constructs a
new array, which is not necessary, and the new array has a different
capacity than the original. It also destroys the data in the other list.

overall: The lines of code to check for illegal indexes in get and remove
are redundant. It would have been good to introduce a private method to
eliminate the redundancy.