1. List the members of your group below. Underline your name.

2. A list of items may be *merge sorted* by recursively merge sorting its left and right halves (possibly in parallel), yielding sorted left and right lists which are then *merged* by repeatedly moving the smaller of the two list heads to the output.

   In more detail, to merge sort a list of items we first divide it evenly into left and right sublists (putting the extra element in the left sublist if the list size is odd). Each sublist is merge sorted recursively, with the base cases of 0- or 1-element lists handled in the obvious manner. The sorted left and right lists are then merged as follows: While both lists are nonempty, compare their first elements and append the smaller one (the left one in case of ties) to the output list. Append any remaining nonempty list to the output.

   In an array-based implementation, both the recursive sort operations and the merging of sorted sublists are easily implemented in place.

   Depict the action of an in-place recursive merge sort on the following array. Depict the state of the entire array immediately after each recursive sort invocation completes.

   $$49, 38, 9, 27, 39, 54, 8, 1, 3, 76$$

[additional space for answering the earlier question]

3. Depict, using class conventions, a red-black tree corresponding to the AA-tree of Figure 19.54 (page 729) of the textbook.

4. Depict the *AA-tree* resulting from the sequential insertion of

$$1, 2, 3, \ldots, 10, 20, 19, \ldots, 11$$

into an empty tree. Depict all intermediate trees. Clearly label any restructuring operations used.

[additional space for answering the earlier question]