P1. Add to your `SortableArray` Java program your implementation of MERGE-SORT as described in CLRS. (You can build on your Homework 1 code or start afresh.) Follow the directions given in Homework 1. Your implementation should differ from mine in two ways:

1. The MERGE procedure should allocate (i.e., instantiate with `new int[...]` where `...` is the appropriate size) two new arrays `L` and `R`, instead of using the pre-allocated array which my code has.
2. The MERGE procedure should use a sentinel. Make sure your test data generator does not generate the sentinel value! Hint: In Java, `0x7FFFFFFF` will give the largest positive int value, since Java ints always use 32-bit 2’s complement binary.

E1. Follow the directions of E1. from Homework 1 to compare the running times of my MERGE-SORT implementation with your implementation of MERGE-SORT from the above problem. More graph paper is attached. Write a short report in English that answers what differences in run time and in what size problems are successfully solved it makes to use the two given different implementation details.

Use only one computer and put in your report the system information listed in Homework 1.

T2. CLRS Problem 2-1 (MERGE-SORT) with larger base cases.)

P2. In CLRS Problem 2-1, `k` could be a constant. Modify my MERGE-SORT implementation so that calls `INSERTION-SORT` from MERGE-SORT whenever the subarray to sort has `k` or fewer elements.

E2. Determine experimentally, as well as you can, the best value for constant `k` to reduce the running time of your implementation of MERGE-SORT from problem E2. above. Just do it for large but reasonably sized problems; the run times should be a few seconds. (You can use shorter times if your computer is so fast that it runs out of memory for problems taking several seconds. Also try to avoid complicating the problem with sizes that make the OS swap memory onto the disk. If you end up working with run times under 1 second, explain in your report why you didn’t use larger problems.)

Do all the experimentation on only one computer; try to keep its load from other programs minimal and constant during your experimentation.

Write a report on what values of `k` you tried, what sizes of problems you addressed, and the running times for the combinations of `k` and problem size you observed. Specify your best single choice for `k` and explain why you think it is the best. Also include the system information listed in Homework 1.