Write an external sort program, with some unit tests. The objective of external sorting is to sort a dataset that does not fit in memory.

To manage this problem effectively in a reasonable period of time, this restriction will need to be artificial. Therefore

A. Write a program which will write N random 4-byte integers to a file (in binary form). Choose N small at first to debug your problem, but it should scale to billions.
B. Write a program which uses an external merge sort to write a sorted version of the file, using no more than M integers of in-memory capacity to do sorting. Start with a small M -- say three -- and make sure it works up to millions. (When M is larger than N, then the sort is no longer external)
C. Write a program which reads the sorted file and confirms that it is sorted correctly.

You may use any library functions your language provides for doing the in-memory sort of the M integers, but you must write the external merge sort algorithm yourself.

Many merge sort algorithms allow large numbers of files to be merged during the merge phase. An ideal algorithm does allow an arbitrary number, but it is acceptable for you to allow only 2 if you like.

Can you adjust your program in the following ways, or, if you do not have time, discuss your strategy for addressing?

1. This program is likely to be I/O bound. Is there any way to take advantage of this?
2. Are there parts of the program that can be parallelized across multiple cores in the same machine?
3. Across multiple disks in the same machine?
4. Across multiple machines?
5. How would one choose M for different N, cores, spindles, and machines?

Please include in your response the URLs of any references you consulted in choosing your external sorting algorithm. Wikipedia is a reasonable place to start http://en.wikipedia.org/wiki/External_sorting