© 2010 Sudarshan S. Chawathe

Name: ____

You should submit (1) a hard-copy of pages 1–6 of this assignment with your answers filled in, and (2) an electronic package that contains the source files for your work on the programming questions, following the submission procedure described below and in class. The hard-copy is due in class before the due date and time, and the electronic package must be submitted before the hard-copy using the interface at `http://cs.umaine.edu/~chaw/u/` only. **No other forms of submission are accepted.**

You are welcome to use any inanimate resources (e.g., books, Web sites, publicly available code) to help you with your work. However, *all such help must be clearly noted* in your submissions. Further, no matter what you use, *you must be able to explain, in detail, how it works.* (You may be called upon to explain your homework in person.) Refer to the class policy for details, and ask for clarifications if you are unsure if something is allowed.

1. (1 pt.) Write your name in the space provided above.

2. (1 pt.) Package and submit your solutions to the programming questions, and any other electronic components of your work, as a *single file* named using the template `cos397-L-F-hw01-N.jar`, replacing `L` and `F` with your last and first names, and `N` with an arbitrary 4-digit integer (and `jar` with `tgz` or another extension if appropriate). You may make multiple submissions (within reason) by using different values of `N`. *After* submitting your work, fill in the following:

File name:	______________________________
Size, in bytes:	______________________________
MD5 checksum:	______________________________
Timestamp:	______________________________

3. (3 pts.) Provide a precise definition of the *scan* operation used by the paper on many-core sorting.[1]. Illustrate your definition with a suitable example.

[1]Nadathur Satish, Mark Harris and Michael Garland, "Designing Efficient Sorting Algorithms for Many-core GPUs," in *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (Rome, Italy, 2009).

4. (8 pts.) Eight students are seated at a round table. In how many different ways can they shake hands simultaneously without any handshaking pair crossing any other? Depict all these ways using a suitable graphical notation.

5. (10 pts.) Repeat Question 4 for ten students.

6. (10 pts.) Describe, in English and as clearly as possible, an algorithm whose input is a positive integer n and whose output is the set of all noncrossing simultaneous handshakes for $2n$ people at a round table. (Thus, Questions 4 and 5 ask for the output of this algorithm for $n = 4$ and $n = 5$, respectively.) Explain why your algorithm is correct.

7. (10 pts.) Provide pseudocode for your algorithm of Question 6, ensuring that the resulting description is clear enough for a colleague to implement without much additional work.

8. (10 pts.) Quantify the running time and memory footprint of the algorithm of Question 7 as precisely as possible as a function of the input. Justify your answer.

9. (50 pts.) Implement your algorithm of Question 7. Instead of graphical output, your implementation should output the set of all possible simultaneous noncrossing handshake schemes as plain text on standard output, one scheme per line. The representation of a handshake scheme on a line follows the template `a-b c-d e-f ...` where the letters are integers in the range $1..2n$ and a pair $x - y$ denotes students x and y shaking hands. There is a single space between each pair thus represented, and the pairs are sorted (on each line) in ascending order of the lower-numbered handshaker of each pair.

   You may write your program in Java or C. (For other language options, please check with me first.) Ensure that your code runs properly on *gandalf*. You must document, package, and submit your source code properly to receive any credit. Do not submit object code or other binary blobs. Ensure that your submission includes a README file with conventional instructions and a Makefile that permits everything to be compiled using a single 'make' command. You should also include at least a few test inputs and outputs. Please ask for clarifications if any of the requirements are unclear, especially because **submissions that do not follow these instructions will receive zero credit.**

10. (50 pts.) Conduct suitable experiments using your implementation of Question 9 to provide an empirical quantification of the running time and memory footprint of your implementation. Compare your results with the answer to Question 8 and explain the agreement or discrepancies.

    Summarize your results appropriately using tables, charts, and text, and include a suitably named file with this material in your electronic submission. In that file, be sure to explain exactly how the experiments were conducted, and how you ensured that the reported results are accurate and significant.