Name: ______________________________

This assignment focuses on (1) solving (algorithm and program) a very simple, but useful, problem; (2) conducting simple experiments; (3) describing the algorithm, program, and experiments clearly; and (4) packaging code and documentation properly. It provides a scaled-down test for similar tasks for Capstone projects. It is also related to the assigned readings, especially the column on literate programming.[1] Further details and clarifications will be announced in class or the newsgroup.

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 `cos497-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:	____________

[1] Jon Bentley and Don Knuth, "Programming Pearls: Literate Programming," *Communications of the ACM* 29/5 (1986).

3. (20 pts.) Describe, in English and as clearly as possible, an algorithm whose input is a set $\{a_1, a_2, \ldots, a_n\}$ and whose output is the set of all permutations of the elements $a_1, a_2, \ldots, a_n$. Explain why your algorithm is correct.

4. (20 pts.) Provide pseudocode for your algorithm of Question **??**, ensuring that the resulting description is clear enough for a colleague to implement without much additional work.

5. (20 pts.) Quantify the running time and memory footprint of the algorithm of Question 4 as precisely as possible as a function of the input. Justify your answer.

6. (20 pts.) Describe the main tasks that you have *completed* in your Capstone project. Use an itemized format and provide specific details, not general comments.

7. (18 pts.) Describe the main tasks that *remain to be done* in your Capstone project. Use an itemized format and provide specific details, not general comments.

   Discuss your answers to this and the previous question with your *project advisor* and ask him or her to *sign below* to indicate agreement.

8. (50 pts.) Implement your algorithm of Question 4. Your program should read from standard input and write to standard output. The input is a set of identifiers separated by whitespace. The identifiers follow the lexical conventions of C identifiers. The output is the list of all permutations of the input set of identifiers, in lexicographic order, one per line. Each line of the output should represent one permutation by listing its element identifiers separated by single spaces.

   You may write your program in Java, C, Python, or Ruby, but without using only the low-level features of these languages (e.g., no Collections framework in Java). This requirement is to ensure the core of the problem is not bypassed. Ask for clarifications, or if you wish to use another language.

   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.**

9. (50 pts.) Present your program of Question 8 in a literate programming style in a PDF file `pg-prog.pdf` that is packaged with your submission.

10. (50 pts.) Conduct suitable experiments using your implementation of Question 8 to provide an empirical quantification of the running time and memory footprint of your implementation. Compare your results with the answer to Question 5 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.

11. (50 pts.) Package and submit all the code and documentation that you have generated so far for your Capstone project. Ensure that the material is well packaged and properly documented. Clearly indicate which parts, if any, are incomplete or not properly working. Include instructions for building and testing your code. Do not submit anything other than *source* code and documentation (plain ASCII text or PDF only). Ask for clarifications.