# Report Guidelines

Larry Leemis, Spring, 2002

Here are a few guidelines for dissertations and semester reports.

## Report Format

1. Have a cover page that includes a title, your name, the date, and a course number, if appropriate. The title should be descriptive and concise.
2. Double space the abstract and body of the text. Single space your reference list, but leave a space between entries. *All* journal article references must contain the author's name(s), year, article title (in quotes), journal name (italics, boldface or underlined), volume number, and inclusive pages. Books must include the author's name(s), year, title, and publisher. There are many different formats for a reference list. Regardless of which one you choose, be consistent.
3. Include a Table of Contents if the paper is long. The titles of the sections should match those in the body of the paper.
4. Number the sections of your paper.
5. Include an abstract that describes the problem, methodology and results.
6. The preliminary pages should be numbered with lower-case Roman numerals.
7. The first page of the body of the paper should be numbered as page 1.
8. Some readers find a nomenclature list helpful. It can be ordered alphabetically or by the order that the items appear in the text.
9. One of the first two sections is typically a literature review to familiarize the reader with what has been done in the past and to assure the reader that you are not reinventing the wheel.

## Figures and Tables

10. Graphical presentation of results greatly improve any report.
11. Label all figures and tables, for example: Figure 1. Mean squared error versus sample size ($n = 5(10)75$). Table titles are placed above the table. Figure titles are placed below the figure.
12. Intersperse figures and tables in text. All figures and tables should be referred to and discussed prior to appearing in the text. They should appear on the page where they are referenced, or on the following page if there is not enough room. The reader should not have to search for a figure or table. Reference figures as Figure 1 (preferred) or Fig. 1, but don't use Figure (1).
13. When a figure is copied from a textbook or journal article, make reference to your source.

## Appendices

14. Place all material that is not appropriate for the body of the text (e.g., a derivation, a proof, computer output, flowcharts, experimental data, or long repetitive paragraphs) in appendices. The results, summary and discussion of the material should be contained in the body of the text. Each appendix should

be referenced in the body of the text and have a title. The purpose of the appendices is to avoid cluttering the body of the report with too much detail. Appendices are typically lettered A, B, C, etc.

15. For statistical tests, show the results in the body of the text and reference an appendix for the details.
16. For computer output in appendices:
    a. All code should be documented by high-level d-charts or flowcharts.
    b. There should be documentation (comments) within the code.
    c. Observe standard coding practices that increase readability (assignment operators and arithmetic operations surrounded by blanks, all statements within a loop or conditional (if-then-else-endif) should be uniformly indented, etc.).
    d. All output should be readable and reduced to 8.5 by 11 inches.
    e. All output should be divided appropriately over report page breaks

**Miscellaneous**

17. No spelling errors. Use a spell checker if you have one that is available. Proofread your work before turning it in. Proofreading is typically the weakest part of the semester projects.
18. Avoid abbreviations.
19. Don't abuse the = sign, e.g., $x$ = waiting time. Instead use: Let $x$ be the waiting time *or* $x$: waiting time.
20. Use three dots only to indicate repetition, e.g., $i = 1, 2, \ldots, n$.
21. Use centered ellipses (e.g., $x_1 + x_2 + \cdots + x_n$) and ellipses placed on-line (e.g., $i = 1, 2, \ldots, n$) appropriately.
22. When a model of a system is used, link the abstract modeling devices to the system. For example, a CREATE block models arrivals of entities that represent unpainted turbine blades.
23. Avoid slang: e.g., *use $x^2$ as an argument in g(y)* rather than *plug $x^2$ into g(y).* Also avoid slang spellings. Use "although" instead of "though"; "through" instead of "thru".
24. Don't use words inappropriately, e.g., *notations, informations, equipments* and *softwares.* Use appropriate verb tenses and be sure subjects and verbs agree in tenses (singular vs. plural).
25. The words *data, criteria, media and phenomena* are the plural forms of *datum, criterion, medium and phenomenon.* Thus, write "data are gathered" instead of "data is gathered".
26. Place two spaces after every period and colon, one space after commas and semi-colons.
27. Keep verb tenses consistent (e.g., Smith (1989) reviewed ...).
28. People's names should always be capitalized (e.g., Poisson distribution, Weibull distribution, Kronecker delta, Newton's laws, Kolmogorov-Smirnov test).
29. Avoid the singular person pronoun ("I") and minimize the use of the plural personal pronoun ("we").

30. Use the "ten or less" rule on numbers. Spell out any number that is ten or less (e.g., "eight observations were collected" rather than "8 observations were collected").
31. Set standard mathematical functions (e.g., sin, log, tan) in roman typeface and single-letter (e.g., $f$, $g$) in italix.
32. Use **boldface** and *italix* sparingly.
33. Use in-line equations and display equations consistently. Also, numbering of equations should be done consistently.
34. Choose one term (hopefully standard in the literature) and stick with it throughout the text. Great creative writers who switch terms throughout their prose make lousy technical writers.
35. Use the hyphen, 1n dash (width of an n), and 1m dash (width of an m) appropriately.

**Programming Standards**

36. Choose meaningful variable names.
37. Avoid the implicit typing feature in languages such as FORTRAN which support it; explicitly declare the type of every variable used.
38. Use structured programming wherever possible (i.e., no goto's) Use if-then-elseif-else-endif and dowhile-enddo statements to avoid using goto's.
39. Use a modular, top-down design approach when writing a program. The main program should be as short as possible, representing only the most general steps of the algorithm. Use subroutines and functions to implement the detailed logic of the program.

    **Example**

    ```
    call read_data
    call run_model
    call calc_results
    call print_results
    stop
    end
    ```

40. Indent two spaces for looping and conditionals

    **Example 1 (branching in C)**

    ```
    if (expression)
      statement1
    else
      statement2
    ```

    **Example 2 (branching in FORTRAN)**

    ```
    if (condition) then
      statements1
    elseif (condition) then
      statements2
    else
        statements3
    endif
    ```

**Example 3 (looping in Splus)**

```
while (condition){
  statements
}
```

**Example 4 (looping in Maple)**

```
for i from 1 to 15 by 2 do
  statements
od;
```

41. Statement numbers begin in column 1
42. Format statements are placed at the bottom of a subroutine
43. Place blanks after commas, around equals signs and around operators if possible
44. Use a consistent character for comments (e.g., * in column 1) and for continuation of long lines (e.g., + in column 6).
45. Avoid type mismatches (e.g., multiplying an integer by a real number).
46. Avoid inefficient programming (e.g., identical statements in *then* and *else* clauses OR placing statements in loops that should be placed before the loop begins).
47. Each program should contain the following information at the beginning (at a minimum):

```
**********************************************************************
*
*  program name:
*  last modification date:
*  purpose:
*  input:
*  output:
*  input format:
*  variable list:
*
**********************************************************************
```

48. *Thoroughly* document all of the code. Each subroutine should include an explanation of its purpose, a description of all arguments and variables and comments throughout the code to explain the logic of the algorithm.