HAVE ASSUPTIONS

E. L. Leiss

Remarks on Programming Style

Basics

- Use structured programming!
- Use procedures and functions! Encapsulate as much as possible. No procedure body (including the main program) may be longer than 95 lines (excluding comments). On average, their length should be about 25-50 lines.
- Do not use global variables! This is an unconditional prohibition.
- Avoid goto's. Programs with more than four goto's are not acceptable.
- Any read must be followed by a write. (You may use a flag to turn off write statements as they mess up your output -- BUT THE CODE MUST BE THERE!!)

Documentation

ANY READ IS FOLLOWED BY A WRITE

(A) Use **mnemonics** for identifiers. Variables have some intended meaning; state it!

(B) In-Line Documentation: For each procedure and function, give a brief description of its input/output behavior, of the parameters used, and if necessary, of some important local variables.

(C) Outside Documentation ✓ INDEPENDENT of THE PROGRAM

A careful statement of the **problem** and a **top-down approach** to the design of your algorithm are mandatory.

You must describe in extensive detail the **input** (including its precise format) your program expects.

You must extensively document and justify any **assumptions** that you made during the development of the program/algorithm (no real-world problem is completely specified). These assumptions must be reasonable. If you have problems justifying them, they may be unjustifiable.

The **data structures** used in your program must be described.

You must not explain the output of your program, since it must be **self-explanatory**.
In other words, if one cannot understand every portion of your output knowing only the problem statement, your program by definition is wrong.
As a general rule, the outside documentation must be logically independent and physically separate from the program. Thus, direct references to the program are not permitted.

Documentation will account for 40-50% of the grade of a properly working program. Thus, a perfectly working program without outside documentation may get a D-.

Thorough testing is crucial. All cases, including exceptions and possible errors must be covered. It is your responsibility to select test cases that convince us that your program works. If we are not convinced by your test cases (not enough of them, many cases not covered, only simple cases covered, etc.), we consider your program not working.
(Remember: Testing means running the program with certain test data and then comparing the output produced with that expected. The only thing worse than a badly tested program is one where the test data handed in prove the program wrong!)