Software Engineering Laboratory Projects and Guides
Version 1.0

Coding Style

General Comments: *The following requirements are the minimum required to ensure correct coding style for support and maintenance.*

These guidelines have been adapted from a coding style guide authored by Prof. Karen Davis, University of CIncinnati.

1. Identifiers

Identifiers shall be descriptive of their purpose or content. Single letter or non-meaningful names are not acceptable. Use correct English spelling, do not haphazardly eliminate vowels. Use abbreviations or acronyms only if they are commonly known.

2. Named Constants

Named constants shall be used rather than literal constants.

3. Modularity

The code shall be written in a modular style. Each module should have only one well-defined task.

4. Variable References

No non-local references to variables shall be made. Variables used in a module are either local or are parameters.

5. Data Abstraction

Good data abstraction shall be employed. The only access to an abstract data type shall be through the procedures or functions that define the behavior of the abstract data type.

6. Partitioning

A complete program shall be partitioned into one or more modules each of which are represented as source code in a separate named file. Each module shall have a module description written as comments at its beginning. The main or root module shall also contain the program code documentation written as comments at the beginning of the file before the module description.

6.1. Program Code Documentation

The format of the program code documentation is:

Purpose:	A brief statement of the purpose of the program
Invocation Syntax:	The specific command syntax for executing the program and definition of each command line option.
Description:	A description of what the program does.
Copyright:	A statement of the copyright including limitations and owner.
Notes:	Any additional notes about the program.

6.2. Module Description

The format of the module description is:

Module name:	Name of the module.
Pupose:	Brief purpose of the module relative to the program as a whole.
Calling syntax:	The specific calling syntax of the module in the form `proc(a,b,c,...) return x` where `return x` applies only to functions.
Inputs:	List of procedure inputs and their definitions.
Outputs:	List of prodedure outputs and their definitions.
Files used:	List and definition of files read and/or written.
Description:	A brief and complete description of the module. If module implements a published algorithm, give reference.
Author:	Name of primary author of the module.
Date:	Date of last update to the module.
Revisions:	List of revision descriptions with latest revision first.

Note that both program and module descriptions are a high-level description of what task the code accomplishes. It should not mention data structures, program modules, or variables. It should be understandable by an intelligent non-programmer.

Correct spelling and punctuation shall be used in all textual descriptions including comments written throughout the code.

The description of program input and output contains the source/destination of all input and output, as well as the formats used for each. BNF may be used for describing formats. A brief example of input and output is often useful.

7. Indention

Adequate use of whitespace shall be employed for indentation and separation of modules. Four spaces for indentation is suggested, and uniform spacing to indicate levels of control flow is required.

8. Line Length

Long lines shall be broken and indented so that they may be easily read on a screen (no longer than 80 characters).

9. File Protection

Unless several students are working on a module, the files containing the module sources shall be set so that no one but the individual programmer may have access to it. For example `chmod 600 filename` gives read/write access to you and not the rest of the world under UNIX.