Project 1 Code Quality Summary - See marked-up code for specifics.

Student:____ **AG score:_______**

General code quality **AG bonus:_______**

____ Function definitions have appropriate comments.
____ Code within functions has appropriate comments.
____ Function prototypes first, functions in readable order.
____ Main function or primary subfunction is compact and conceptually simple.
____ Code is not duplicated excessively.
____ Program has well-chosen subfunctions to organize the code.
____ Program has no problems with "Swiss Army" functions.
____ Program lacks redundant, convoluted, or awkward code.
____ Code is clear and easy to read: not obscure, verbose, or excessively nested.
____ Good variable/symbol naming and usage.
____ Program appears to be free of egregious inefficiency.
____ Good choice of functions in utility module (e.g. a string allocator/deallocator).
____ Standard Library facilities used appropriately (no recoding of wheels).
____ assert macro used to clarify code and help detect programming errors.
____ Program provides single points of maintenance for program parameters.
____ C used idiomatically, following K&R and lectures.
____ (can be negative) Code follows guidance on other matters in C Coding Standards and course material.

Specific code quality - following specified and recommended practices.

code structure and safety practices

____ Command handling uses a switch statement that calls command-specific functions.
____ File input reading loops are correctly structured.
____ Files closed shortly after last input or output operation.
____ Input of strings disallows overrun of array.
____ #define used to avoid "magic" numbers and strings embedded in code.
____ Global variables are only read in p1_main.c, modified in responsible modules/functions only.

memory management

____ Container, Record, Collection memory allocated and deallocated in each module's functions only.
____ String memory allocated and deallocated, and global variable maintained in separate functions (e.g. in Utility).
____ Memory for strings allocated to fit the data (strlen + 1).
____ No unnecessary memory allocations (e.g. for temporary buffers - local arrays used instead).
____ Return value from malloc checked and program terminated if failed.
____ Program appears to free all allocated memory (no leaks, even at termination).

headers and linkage

____ C Header file guidelines followed (e.g. no unneeded #includes in .h files).
____ Internal linkage specified for "helper" functions, or not needed.
____ Global variables declared (only) as extern in p1_globals.h; defined in .c; .h declarations used throughout.

Other problems

____ (0 or negative) Program has additional problems:

____ **Total**