**CS 499 Code Review Checklist**

**Structure**

❏ Does the code completely and correctly implement the design?

Yes

❏ Does the code conform to any pertinent coding standards?

The C code no, but the Python code conforms to PEP-8

❏ Is the code well-structured, consistent in style, and consistently formatted?

Yes

❏ Are there any uncalled-for or unneeded procedures or any unreachable code?

No

❏ Are there any leftover stubs or test routines in the code?

No

❏ Can any code be replaced by calls to external reusable components or library functions?

No

❏ Are there any blocks of repeated code that could be condensed into a single procedure?

No

❏ Is storage use efficient?

Yes

❏Are symbolics used rather than “magic number” constants or string constants?

There are definitely some magic numbers but it’s minor and helps with clarity

❏ Are any modules excessively complex and should be restructured or split into multiple routines?

No

**Documentation**

❏Is the code clearly and adequately documented with an easy-to-maintain commenting style?

Yes

❏Are all comments consistent with the code?

Yes

**Variables**

❏ Are all variables properly defined with meaningful, consistent, and clear names?

Yes

❏ Do all assigned variables have proper type consistency or casting?

Yes

❏ Are there any redundant or unused variables?

No

**Arithmetic Operations**

❏ Does the code avoid comparing floating-point numbers for equality?

Yes

❏ Does the code systematically prevent rounding errors?

Yes

❏ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?

Yes

❏ Are divisors tested for zero or noise?

No, but no division is being performed

**Loops and Branches**

❏ Are all loops, branches, and logic constructs complete, correct, and properly nested?

Yes

❏ Are the most common cases tested first in IF- -ELSEIF chains?

Yes

❏ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?

Yes

❏ Does every case statement have a default?

No case statements

❏ Are loop termination conditions obvious and invariably achievable?

Yes, except for deliberate infinite loops

❏ Are indexes or subscripts properly initialized, just prior to the loop?

Yes

❏ Can any statements that are enclosed within loops be placed outside the loops?

No

❏ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Yes

**Defensive Programming**

❏Are indexes, pointers, and subscripts tested against array, record, or file bounds?

No

❏ Are imported data and input arguments tested for validity and completeness?

Yes

❏ Are all output variables assigned?

Yes

❏ Are the correct data operated on in each statement?

Yes

❏ Is every memory allocation deallocated?

No, but an infinite loop is presupposed

❏ Are timeouts or error traps used for external device accesses?

Yes

❏ Are files checked for existence before attempting to access them?

For python stuff yes, for C stuff no

❏ Are all files and devices left in the correct state upon program termination?

Yes