# Checklist for CS 211 Code Reviews

## Structure

□ Does the code completely and correctly implement the design?
□ Is the code well-structured, consistent in style, and consistently formatted?
□ Are there any uncalled or unneeded procedures or any unreachable code?
□ Are there any leftover stubs, test routines, or debugging statements in the code?
□ Are there any blocks of repeated code that could be condensed into a single procedure?
□ Does the code use overly complicated logic or methods that could be simplified?

## Documentation

□ Has the file been given a meaningful name, separate from the given base code file name?
□ Does the program begin with a comment giving the author, date, and purpose of the code?
□ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
□ Are all comments consistent with the code?

## Variables

□ Are all variables properly defined with meaningful, consistent, and clear names?
□ Are all variables properly typed for the information they are to hold?
□ Are all variables initialized or otherwise given a value before they are used.
□ Do all assigned variables have proper type consistency or casting?
□ Are there any redundant or unused variables?

## Memory Management

□ Is all dynamically allocated memory freed before program exit?
□ Are all pointers initialized to NULL, and reset to NULL when no longer in use?
□ Are memory structures sized appropriately for their use?

## Arithmetic Operations

□ Does the code avoid comparing floating-point numbers for exact equality?
□ Are divisors tested for zero?
□ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?

## Loops and Branches

□ Are all loops, branches, and logic constructs complete, correct, and properly nested?
□ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
□ Does every switch statement have a default?
□ Are loop termination conditions obvious and invariably achievable?
□ Are indexes or subscripts properly initialized, just prior to the loop?
□ Can any statements that are enclosed within loops be placed outside the loops?
□ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

## Defensive Programming

□ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
□ Are imported data and input arguments tested for validity and completeness?
□ Are all output variables assigned?
□ Are the correct data operated on in each statement?