CSDS 325/425: Computer Networks
Programming Project Guidelines

Guidelines for All Projects:

- All projects *must* be completed in C or C++.

- All submitted projects will be graded on the departmental Unix facilities. You are responsible for ensuring they compile and run correctly in this environment. *If you develop elsewhere, start the process of trying your project on the class server early.*

- All projects *must* conform to the guidelines given in the assignment (e.g., in terms of including a Makefile, executable name, output format, command-line options, etc.) to ensure we will be able to automate portions of the grading. Violating the project specifications will result in a grade deduction.

- Violating the output format specifications for a project will result in at least a 10% grade reduction.

- You will not be graded on optimality. However, you are expected to turn in reasonable programs. Hence, massively suboptimal programs—in terms of runtime and/or memory usage—*will reduce your grade.*

- Projects that do not exhibit good programming practice—as sketched below—will result in grade deductions.

Good Programming Practice Touchstones:

- While there is no prescribed style you must use, you should use a consistent style throughout each program (e.g., indentation, bracket placement, use of mixed caps, etc.).

- Use meaningful variable names.

- Define constants and do not use literals inside expressions (i.e., use constant variables or `#define` macros).

- Use proper modularization (functions, methods, classes, etc.). Methods should not be overly lengthy or try to do too much.

- Comment anything that might be a bit tricky to understand.

- Use data structures appropriate for the given task. You will not be graded on optimality—and there is often not one *right* answer given that there are engineering tradeoffs at play—but grossly suboptimal code will result in a grade reduction.

- Check return values and make sure your programs behave reasonably in the face of errors.

- Do not waste memory (e.g., remember to *free()* memory you no longer need, etc.).

- Do not perform needless computations and remember computations that are used frequently.

- Ensure that your programs perform well with large input sets. Do not think "bytes and kilobytes", think "megabytes and gigabytes".