Coding Standards

CS240 Programming in C
Coding Standards

Although you will not be graded on it, coding style is still very important for the readability of your code. Bonus, it makes testing easier, too. There are 3 main things you should keep in mind when writing your code.

● **White Space, i.e. indent your code.** You should indent your code whenever you enter curly braces.

● **Modularization**, i.e. break your code up into small chunks, making each its own function. Test each of these as you finish them, as this will save you much time later.

● **KISS**, i.e. Keep It (sorta)Short and (somewhat)Simple. You should try to only write the code you need, it should be beautiful and the purpose should be immediately obvious. You could write a lot of code(modularized and unit tested, of course), then work on combining statements and minimizing lines (see followup discussion). You should strive for a good balance of simplicity and brevity.

NOTE: This is simply for the aesthetic value of your code, practical coding habits should also be kept in mind, exempli gratia meaningful variable/function names, using #define's instead of "magic(hardcoded) numbers", good use of comments(brief, descriptive, not redundant), etc.

Following these tips, along with taking your time and not coding hastily, will help you code and debug more effectively and efficiently, and it will also help you better understand what your code is doing.

May the code be ever in your favor.

For projects, bad coding style may cause you to lose points. If you feel the article http://www.doc.ic.ac.uk/lab/cplus/cstyle.html is too long to read, here is a short list:

● Do not use hardcoded values (magic numbers); use defines or const (names for constant literal should be of capital letters)
● Variables should be initialized if initial value is important, or add a comment if the code relies on their initial value being 0.
● For pointer, the * should be next to the variable and not next to the type name.
● E.g., char *ptr instead of char* ptr.
● Always check the return code of any function that can fail, including malloc or opening files, even if we allow you to assume the function always works on Autograder.
● Proper work with dynamically allocated memory: check pointers, set pointers on null after freeing, etc.
● Proper use of names for functions and variables, names that are meaningful.
● Comment your code when needed
● Use consistent indentation and group related lines together to ensure more readable and clean code.
● When functions are in the same file as main, prototype of functions should be before main and body of functions after main.
● Do not use same names for global and local variables.
● Using local variables is preferred most of the time; however, global variables are fine in some cases. Variables that are required throughout the life of your program should be declared in main and passed to other functions as needed. In general, smaller scope for variables is better because less code can potentially modify the variable, thus solution is less complex. If you must use a global variable, always ensure it is only modified directly by one method and have all other callers use that method. This will make debugging easier.
● We will update the list as the course goes forward to reflect new problems we find.

About comments
Comments are helpful for other people reading your code. if you write "this function takes an int and returns an int" then that isn't helpful, because the reader can already tell by looking at the code, and they still don't

know much about your function. A better comment for functions would include what each argument represents and what the return value represents, as well as any conditions that must be met when calling the function.

The comments might also depend on where you are commenting. In a header file, your comments should be geared toward people using the function. In a source file, your comments might be geared toward someone editing the function.

However, for labs and projects, you can get away with just gearing your comments towards the graders, so we can tell what your code is doing.