Programmer's Checklist:
(last updated Wed Nov 4, 2009)

Remember, *programming is easy, design is hard*. Time spent on design is time spent well, and will make programming go much faster and yield much better results.

The following are some items for you to check before you submit your program. Some are high-level, others are more detail oriented - all will influence the quality of the final product and the score you will earn. Hopefully they will be helpful to you in writing better programs.

Document your program. Someone other than you will eventually look at it; don't have them guess your intentions.

Main program – is it modular? Does it read like an outline of your general algorithm?

Use of functions – do your functions perform a clearly defined task? Do they only use parameters and local variables to accomplish their mandate?

Visibility – always favor private over public.

Is your code formatted consistently according to common conventions? Break long lines, a programmer's editor is not a word processor.

There is elegance in simplicity – don't make your code more complex than it needs to be.

Choose the appropriate techniques and data structures - Recursion vs. iteration? Arrays vs. linked lists? Be able to justify your choice.

Global variables – if you have them your code will be less re-usable and unless you can carefully justify their use (document this) you should avoid them as much as you can.

Pointers - if a function receives a pointer (to a "string" or a data structure, like a linked list) check *first* to make sure it's not NULL *before* you do anything else.

Resources - if you are allocating memory (malloc/new/..) .. be sure you free it up when you are done with it. Same for files, any files opened need to be closed. This will ensure that all buffered data is written out.

Numeric literals – there is no good reason to have any numeric literals in your program. Use constant symbols instead, they will make your code more readable, and usually also much easier to modify.

Error messages – they should be as specific as possible (so they can be helpful to the user) and should always be directed at stderr.

Testing – test a range of normal values for basic functionality, followed by boundary values and no values. (Many an interactive program has crashed when presented with a simple <CR>, or an empty file)

All of the above are taken into consideration during the evaluation of programs. Much more could be said about any of them, feel free to stop by to discuss these. Some of these are a matter of degree, let your instructor's experience guide you: he has designed and implemented code for longer than you and he gets to evaluate your work – two good reasons to consider his input.

EB