

CSCI 320 General Policy for Programming Assignments

Fall 2004

DUE DATES

To receive full credit, each programming assignment must be handed in **on the due date no later than the closing time of the CSCI labs**. If you skip class on the day a program is due, it is automatically one day late even if handed in later that day.

Programs may be turned in after the due date for partial credit. You will lose up to 5% of the total points for each day the program is late (a day is a school day MTWRF, not including weekends or holidays). No program will be accepted for credit after the graded programs for that assignment have been handed back.

PROGRAMMING STYLE

Each program you turn in **must** contain an initial block of comments with this information:

- 1) your name
- 2) the class
- 3) the assignment number
- 4) the C++ compiler you are using if it is not Microsoft Visual Studio .NET 2003
- 5) the main function file should name **all** of your source files needed to run the program
- 6) a detailed statement of the purpose of the program -- assume the reader has not seen the program assignment and needs a general understanding of what your program does and how it works

The program should contain sufficient additional comments to clarify what you're doing. A reader of the program should be able to determine from the comments alone a general idea of what is happening without actually having to decipher any code. Avoid redundant comments which don't add to the reader's comprehension. Good comments explain **why** you're doing something or what it means in the context of the program.

Here's an example of what I consider to be helpful comments in code searching for the largest value in an array. Also notice how the variable names are chosen to describe the values the names represent.

```
hiVal = arr[0];           // Assume first element's value is largest
hiSub = 0;                // Remember position of largest value
n = 1;                    // Start searching with the second element
while (n < MAX)           // Look at each element
{
    if (arr[n] > hiVal)    // If this element value is larger than previous largest found,
    {
        hiVal = arr[n];   // save it as largest,
        hiSub = n;        // and remember its position
    }
    n++;                  // Look at the next element
} // end of loop to find the largest element
```

Modularize your programs. The main function should have minimal references to implementation-dependent details. Unless a particular assignment states otherwise, **only file variables may be declared globally; all other variables used by a function must be declared locally or passed as parameters**. Every program module should have a block of comments explaining what the module does.

Make your classes general and flexible for maximum reusability. Provide class access functions to return private variable values so that clients can code their own functions for operations the class implementer didn't anticipate.

Choose reasonably descriptive identifiers to name variables, constants, functions, classes, etc. Single-character identifiers are unacceptable except for variables like subscripts and loop-control variables (and a more descriptive identifier is recommended for these variables whenever it is appropriate).

Please use at least minimal indentation standards such as indenting statements within a loop, aligning the clauses of IF statements, etc. This will not only make me happier but will also make your life easier in reading and debugging your own programs. Following the coding styles used in the Nyhoff book programming examples would be one way of choosing a good coding style. In any case, whatever style you choose be consistent in applying it throughout your code.

Use blank lines liberally to separate sections of your program, and use extra spacing within a statement to enhance readability.

HAND IN

For each assignment, hand in a machine-readable copy of all source files needed to run your program. Since I will compile and run each program myself, it is not necessary to give me other files, like workspace or executable files.

Your main program should have a filename of your name or initials and the program number (for example: SMITH1.CPP or JMS1.CPP or JIM-1.CPP)

You may:

- 1) Hand in a disk (which will be returned to you as soon as I copy the files).
The disk should contain ONLY the files for this assignment.

**Make sure you have backup files -- NEVER give me your only copy.
Don't forget to label the disk with your name and the class.**

or

- 2) Send the file(s) by email (highly preferred) to: Derek_Harter@tamu-commerce.edu
Please include in your subject line your name, the class, and the program number,
like: Jim Smith 320 Program 1

No matter how you hand in your program, please make sure that your name appears in comments in **every** file, including output files created by your program and class files taken from the text.

PROGRAM EVALUATION

Each program will be graded on a 10-point basis:

- 0-1 Little or no original work accomplished
- 2-3 Program doesn't compile, but you've made a reasonable attempt
- 4-5 Program compiles but produces little or no meaningful output
- 6-7 Program compiles and runs, but there are major errors
- 8-9 Program compiles and runs, but there are minor errors
- 10 Program runs correctly, satisfies all requirements, and has good programming style

PLAGIARIZING

Programs which are so similar that they are almost certainly copies of one another will receive a grade of zero. Unfortunately this penalizes the original author as well as those who cheat. To protect yourself from this penalty, protect your own work. Don't leave copies of your program on the hard disk of a lab PC and don't leave diskettes or hard copies of your program lying around. Definitely do not give anyone else a copy of your program.

MINIMUM REQUIREMENTS

You must have at least a passing (60%) average on programming assignments in order to pass the course, regardless of how high your exam grades may be.

Tips for successful programming:

- Spend some time on the design of your algorithm and consider possible alternatives before you start coding.
- Design your program in modules, no matter how you actually code it.
- Code and test one module at a time.
- Always echo your input to the screen to make sure your data is being read properly. Display intermediate results as you test each module to be sure it's working as you want it to.
- Don't wait until the last minute to start working on a programming assignment.