1 Grading

Lab exercise parts (such as part2 of lab2) which can or must be “turned in” for grading and credit will be graded on a “binary scale”: Did you do what was specified for credit, or not? “Turned in” lab parts will be due midnight Friday of the week after the week the lab was conducted. If classes are not in session Friday, it will be due the following Friday.

The projects are different from the lab exercises. They are really “projects”. That means they entail analysis of the problem parts, learning topics relevant to their solutions, planning for incremental development, design, coding, testing, debugging and the solution of unanticipated problems spread out over the two or so week period over which the project is assigned. There will be pitfalls and problems to solve that you will not know about until you work through the project details! Most students who do no substantial work on a project before a couple of days before the due date will encounter frustration and failure.

Projects submitted by 11:59PM of the first due date are not late. Projects submitted more than 48 hours from that time receive no credit, but may be taken into consideration by the professor when deciding marginal grades. Projects submitted in between those times will have their grade multiplied by the following lateness factor:

\[
\text{LatenessFactor} = 100\% - \frac{\text{TimeSubmitted} - \text{TimeDue}}{48\text{hours}} \times (50\%) \]

This means roughly 1% will be subtracted for each hour late. The calculation will be done by computer so accurately (< 1%) that roundoffs will make no difference in your final letter grade.

Project grade reports will be emailed to your itsunix Unix account. Consult the helpdesk or online web material for help to make sure you will receive this email. Don’t ask us to email such reports to other addresses specially because the answer will be no.

The Projects will be graded as follows:

- A part that does not compile and link to an executable file that can be run gets no credit. Never make any change, not even a comment, between testing and submission. See the advice for final testing in the section on submission. Prof. Haas has seen seen people turn 100% into a zero by ignoring this rule, and I will follow this lead.

- 70% for run time performance. This includes generality of the inputs, exception handling, ease of use, and (of course!) the program has to run. If you cannot get the full program to run, concentrate on putting to work smaller versions of the problem (i.e. with some the function bodies being stubs, or some test commands printing a simple “Not implemented.” first), instead of having a piece of code that doesn’t do anything (you will get some partial credit instead of no credit at all).

- 30% for programming style (reasonably consistent and clear indentation), structure (use of specified separate header, template and implementation files following the practices given in
Main and Savitch’s textbook), revision control (submission of reasonably complete revision history in the form of RCS database files for each non-trivial source file) and comments in the form of precondition, postcondition and invariant statements. Other kinds are comments are optional, and are encouraged if they will help you do the programming and design.

2 Cheating

Your programs are supposed to be entirely your own work. You may discuss problem and programming issues with others, but the code you submit must be written and typed in by you, except that you may download and modify test driver and header files from the textbook’s or this course’s web site: NO OTHER WEB MATERIAL MAY BE COPIED.

If two programs look to me like they are too much alike to be a coincidence, both get zeros. We use a computer program to compare homeworks, then check by hand to be sure. The absence of a credible RCS revision history will provide additional evidence for a charge of cheating. A second offense gets an E in the course. That is not an empty threat – Computer Science professors do it several times a year, and some students never graduate because their major grades have been lowered because of cheating. All cases of cheating are reported to the Office of Undergraduate Studies according to the policy on page 29 of the 2002-3 Undergraduate Bulletin.

Everybody’s account is initialized so nobody else can read its files. Do not change the protections (unless it’s of home and public_html directories to make a web page, and you know what you are doing there): Your homework may become world-readable. It is your job to keep your work safe from prying eyes.

3 Turning in Project and Lab Work

1. Very Important (for your grade!!) Do a final test that your build.sh script builds the submitted software without any errors or warnings. Then, do a final test of its functionality.

   It is our experience that SPARC Solaris makes more “garbage” values in uninitialized dynamic and local variables (which standards conforming C/C++ are allowed to do, for efficiency reasons). Therefore, we have seen many student C/C++ programs that apparently work under IA-32 Linux to fail under SPARC Solaris.

2. Very carefully remove all executable and object files from all the directories (the specified one on down) you will submit. What is safer is to plan ahead and create a cleanup.sh script that automates the proper rm commands. That way, will you avoid the mistakes you are more likely to make when you do an unusual thing when you are under stress. (Yes, I know what kinds of course work here is stressful.)

   The object and executable files must be removed so the class account storage quota is not exceeded. Penalties will be subtracted if we receive them.

3. Note the project or exercise turnin name from the assignment. Suppose for illustration’s sake that the turnin name for the project or exercise is “lab2”. (This is the actual turnin name for the first lab exercise of Spring 2003 CSI310.) When you are ready to submit your work, go (use cd) to the directory directory-name ABOVE the one you plan to submit. Type the command
turnin-csi310 -p lab2 -c csi310 directory-name
where directory-name stands for the directory to be turned in. (Seek help if the command wasn't found: It's in /usr/local/bin)

The system responds with
Your files have been submitted to csi310, lab2 for grading.

Please note that using the turnin program as above is the ONLY acceptable way of submitting programming assignments in this course. You should NOT mail the files to the instructor or the TAs.

4. Pay attention to all error messages printed by turnin: An error message probably means your submission will not be received. Recheck the instructions, repeat until turnin use with −v described below shows successful receipt, and see a TA if positive acknowledgment is not obtained.

5. If you use the turnin command above again at a later time, then everything you submitted previously will be replaced by the newly submitted files. (This allows you to resubmit a program if the previous submission was erroneous. But it wipes out the submission time record of the previous submission, so your revised submission will be more late.)

6. If you have a good reason you want to turn in a revised or corrected version of your work without wiping out the record of your previous version, get permission from the professor or any TA to turn in the revision to an alternate “-extra” project. Permission generally will be given, but work in the alternate turnin will be penalized for some lateness.

7. Strongly Recommended!! After any submission, run
   turnin-csi310 -c csi310 -v -p projectname
   and observe the report of what files were received. If the file(s) you meant to send are not listed, check and redo the instructions. See a TA if you cannot get a positive acknowledgment.

$Id: turnin.tex,v 1.5 2003/02/12 03:36:20 sdc Exp $