CSI 310 – Data Structures
Lab and Project Grading Policies
Instructions for Turning In Lab Exercise and Project Directories

1 Grading

Training exercise “checkoff credits” will be awarded on the basis of the files and directories you “turned in” as instructed in each assignment, plus one credit for attendance during the week of that exercise. Each “checkoff credit” will be graded on a binary scale: Did you do what was specified for credit, or not? “Turned in” lab material parts will be due midnight Friday of the week after the week the training was assigned. If classes are not in session Friday, it will be due the following Friday.

Checkoff credits will also be awarded for satisfactory performance on quiz questions plus the lecture attendance manifested by your submitting a quiz paper. Quizzes will be given at unpredictable times.

Independently of the “Training Exercises” Your lab/discussion TA may specify material to be turned electronically for lab credits. He or she will tell you the project name to use at such times.

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.

The due dates for each project will be specified as a pair of dates, like Feb. 8-10. Projects submitted by 11:59PM of the first due date are not late. Late projects submitted after the end of the last date provide no credit towards your project score, but may be taken into consideration by the professor when deciding marginal grades. A project submitted during the “grace period” between those times will have its grade multiplied by by the following lateness factor:

\[
\text{LatenessFactor} = 100\% - \frac{\text{TimeSubmitted} - \text{TimeDue}}{\text{Time within grace period}} \times (50\%) \]

For example, for the dates Feb. 8-10, the grace period is 2 days (48 hours); the formula 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. (The penalty rate of 25% per day may be reduced and the grace period of 48 hours increased for the longer projects later in the semester.)

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.

The Projects will be graded as follows:
• Projects will often consist of two or more “parts”. For each part, all the sources must be included that will produce one executable program when they are compiled and linked by a “build script” (see below) that must also be included. From the beginning, each part must be built from several different header and C++-function files. Each part will be counted separately.

• 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 and I have 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 consistant 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 or other MATERIAL MAY BE COPIED. You may also copy specified project material from the course account public directories under “acsi310 on the ITSUNIX cluster file system.

If two programs look to me like they are too much alike to be a coincidence, both get zeros. We may use a computer program to compare homeworks, then check by hand to be sure. The absence of a credible RCS revision history, or a forged 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 (or Graduate) Studies according to the expectation of faculty under the “Penalties and Procedures for Violations of Academic Integrity” in the Albany 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**. Ideally, a full regression test (using ALL test cases used in development and in debugging) should be done next. Finally, **extremely important**, carefully make sure there are no “core” files (named “core” and typically a few megabytes long) in the directories you are going to submit.

   It is our experience that SPARC Solaris, compared to Linux, has more “garbage” values (fewer zeros) in uninitialized dynamic and local variables. (Standards conforming C/C++ systems are allowed to leave such variables uninitialized, 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 “core” files, objects and executable 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 core, object and executable files must be removed so the class account storage quota is not exceeded. Penalties will be subtracted if we receive them, especially for the core files.

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 “Train1”. (This is the actual turnin name for the first training exercise of Spring 2006 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

   ```shell
   turnin-csi310 -p Train1 -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

   ```text
   Your files have been submitted to csi310, Lab1 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 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.
An error message indicating a quota is exceeded is usually caused by other students having submitted core files. If you get this message, email the professor and TAs immediately. We will investigate as soon as possible, fix the problem, and get very angry at all the students who submitted core files.

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 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 send the revision to an alternate “-extra” project. Permission generally will be given, but late version will be penalized for some lateness.

7. If you run into a programming jam, you can email the professor (sdc@cs.albany.edu) with a message describing the specific problem, and if you wish, a note indicating that you have turned in ALL THE FILES one needs to compile enough to reproduce the problem. Do the turnin to project debug.

I try to respond to email in a reasonable time, but make no guarantees. I will also ignore messages that are general complaints or not specific enough for another person to provide any help.

In particular, you are expected to carefully read every line in the project assignment, the published C++ function and header files for the project, relevant text readings and lecture materials from the Web. A request for clarification of the assignment will be rejected or ignored if it doesn’t cite the page, paragraph, or line where you think there is an ambiguity, error or need for clarification.

8. 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.