Grading Guide for the Notes for Lab 2

Guidelines

Joint notes for each lab should contain all the information you need in order to write your lab report. Some tips:

- **Good software engineering.** Only make ***minimum*** changes to fix defects. Do ***not*** change things that are not defects. Write clear code: Be consistent in following fixed conventions for indentation etc.
- **Copy and paste.** The idea is to record what you ***see*** and what you ***do***. Copy and paste input, output, changes to the code, and so on. Err on the side of including data rather than discarding it, but if you get many compiler errors at once, it is sensible to copy only the one you set to work on immediately, then compile again and repeat. Copy and paste old and new lines of code; writing just line numbers is not clear enough.
- **Include input files.** When you create an input file to test your program, include the contents in your notes. This can be done simply by running `cat` on the file and copying and pasting the entire command and output.
- **Record as you go.** Don't wait until you've fixed a bug to start recording detail. Record what you do and observe as it happens, whether it is what you expect or not.
- **Don't write too much.** Write summaries for observations you cannot copy and paste, such as "after 10 seconds the program still didn't terminate." Also write brief notes for clarity, especially to explain code changes, but these need not be complete sentences. Too much writing will slow you down, and it is important to finish the lab before 7pm, ideally much sooner.
- **Include enough test cases.** Make your own ***additional*** test cases. When you think you have finished debugging the program, re-run all your test cases to demonstrate that the program meets the specification.
- **Include your final code.** For short programs this can be helpful; but programs that are long or made up of many files should be kept separate from your notes. We will let you know when you do not need to include the final code.
- **Be understandable.** The reader must be able to follow what you did easily. Graders will take points off for lack of clarity. Use standard English, as opposed to colloquial language like "TOTAL FAIL!!!"

Grading Rubric (maximum 15 points)

Category	Characteristics of excellent notes (+5)	Characteristics of acceptable notes (+3)	Characteristics of notes that need improvement (+1)
Detail	A. Includes raw input and output. B. Clearly indicates code changes. C. Includes test file contents.	D. English writing mistakes. E. Some steps are described ambiguously or inaccurately.	F. Describes most steps ambiguously and/or inaccurately G. Not enough detail to get a general sense of what happened.
Completeness	H. Includes all steps, working and not. I. Includes sufficient test cases. J. Includes final code. K. Maintains chronological order.	L. Includes most steps, but not all. M. Shows some test cases. N. Includes final code. O. Maintains chronological order.	P. Omits many steps. Q. Does not include test cases. R. Omits final code. S. Does not maintain chronological order.
Correctness	T. Fixed all bugs.	U. Missing or incorrectly fixed one bug.	V. Missing or incorrectly fixed two or more bugs