Code Tracing  
CSE8B Spring 2009

Code tracing is a form of analysis in which a human being (usually using pencil and paper) acts out the roll of the computer executing a program. In the real world, this is often done

- in order to read and understand a piece of code (which may be part of a much, much larger system and could be hard to isolate when running the whole program), OR
- to debug a piece of code which is not working correctly (again, tracing “by hand” is often an easier first step when code is part of a large system)

In your education, tracing is something that you should do on exams and other pencil and paper assessments. It may be something you are asked to do directly (because it illuminates very clearly your understanding of the code). But even if you aren’t, tracing will often keep you from making small mistakes that might cost you points. Including trace tables on your exam may also help you earn partial credit.

**Tracing Basics:**

- Follow, line by line, the DYNAMIC execution of a piece of code (following all conditional and iterative statements). Most often, tracing centers around loops.
- Keep track of the values of variables as they change. (Sometimes only recording a subset of “changing” variables).
- Do so in an organized, physical, written-down form to support reflection, review/checking, and analysis.

**When should I do a trace on an exam:** Consider doing a trace when you see questions of the form

- What is the value of x, foo, or grades at the end of this code
- What is printed when this code executed? How many times does the print on line 12 happen?
- Trace the values in variables foo and bar on the following input...
- This following code is buggy and may not do what the programmer intended (what is value of foo, fix the code so that it correctly finds the average)

**When should I trace code in “real life”:**

- When your code doesn’t perform as expected and there are 4 or fewer key variables to trace
- When running the code to debug it will take a while to set up (or you cannot reproduce the bug on your system easily, or it is hard to get output to “show off” the bug in question, when you must run a really large application and the suspected buggy part is hard to identify during runtime)
- When you are trying to read and understand code written by someone else (or you, last year) so that you can modify it or fix it