

CSE306 Software Quality in Practice

Dr. Carl Alphonse
alphonse@buffalo.edu
343 Davis Hall

opaque testing

- Tests are written without regard to **HOW** code is written



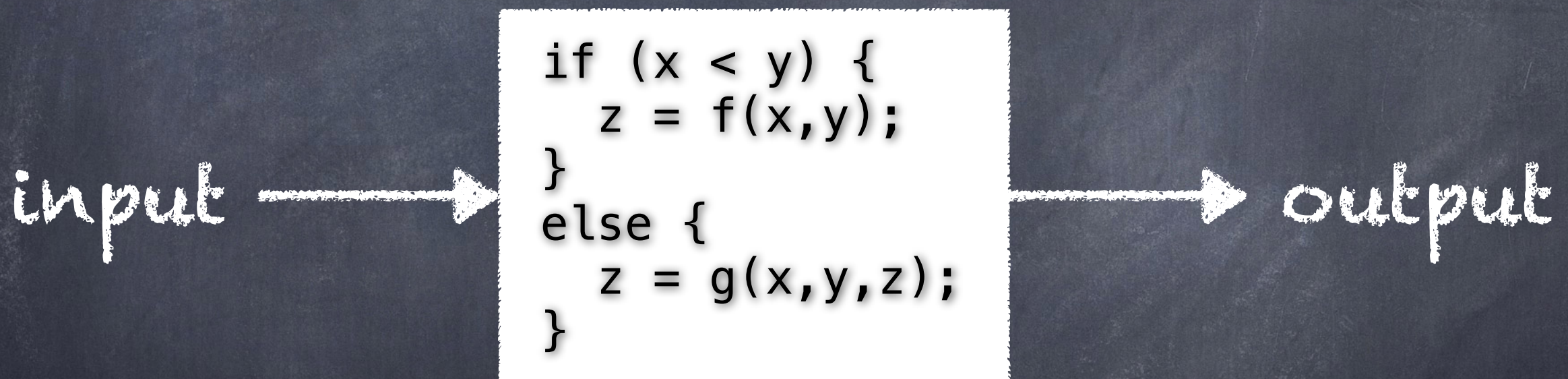
opaque testing

- Tests are meant to capture the intended behavior of the system (the requirements/specifications): **WHAT** the code should do.



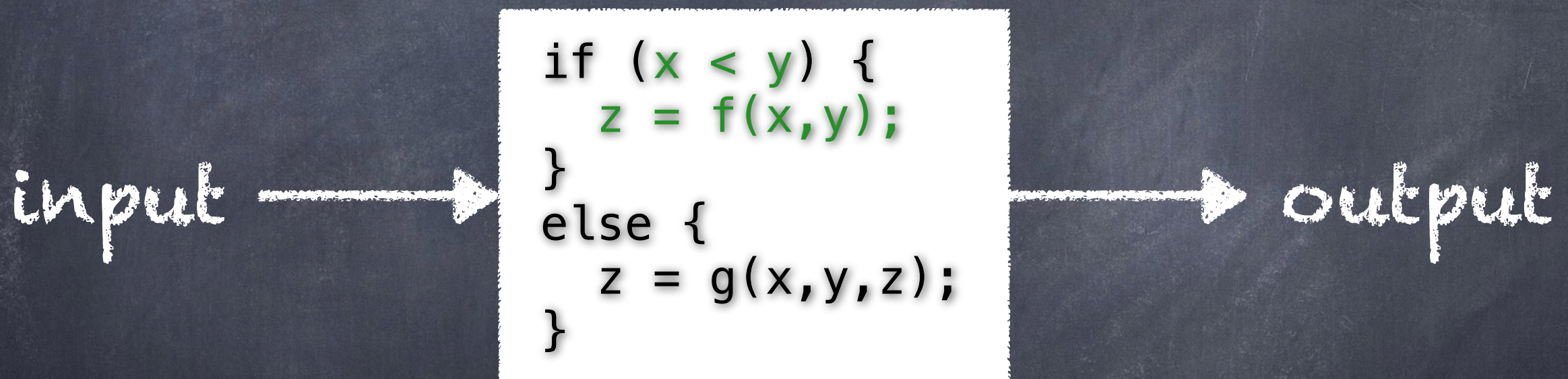
transparent testing

- Tests are written taking into consideration **HOW** the code is written.



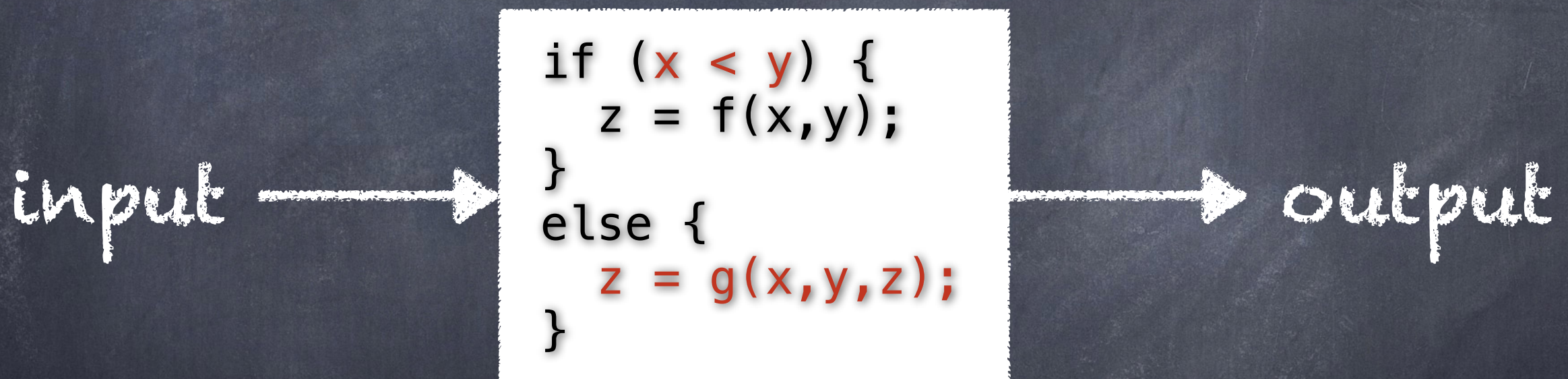
transparent testing

- Use a code coverage tool to ensure that tests exercise **ALL** possible computation paths.



transparent testing

- Use a code coverage tool to ensure that tests exercise **ALL** possible computation paths.



Code coverage

- We will use gcov as our coverage tool.
- Compile with,

```
-fprofile-arcs  
-ftest-coverage  
-lgcov
```

- as in:

```
gcc $(CFLAGS) -fprofile-arcs -ftest-coverage  
-L /util/criterion/lib/x86_64-linux-gnu  
-I /util/criterion/include  
$(OBJECTS) tests.c -o tests  
-lcriterion -lgcov
```


using gcov to verify test coverage

- ◉ compile test code with extra flags
 - ◉ this instruments code to gather coverage information
- ◉ run tests
 - ◉ this runs your tests and allows the instrumentation to collect coverage data that shows what parts of the implementation were exercised by the tests
- ◉ run gcov on the source file (e.g. source.c) whose coverage you're interested in exploring
- ◉ use 'man gcov' to see gcov command line options. Try -b.
- ◉ Look at the file produced by gcov (e.g. source.c.gcov)

Lecture question

Exercise:

[https://tools.ietf.org/html/
rfc3986#section-3.1](https://tools.ietf.org/html/rfc3986#section-3.1)

(GH Classroom link on
course website)