blackbox vs whitebox testing
blackbox testing

Can anyone describe what this is?
blackbox testing

Code is treated as a “black box”, one which you cannot peek inside
blackbox testing

Tests are written without regard to how code is written

input → blackbox → output
blackbox testing

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

In Test Driven Development (TDD) tests are written before the code is, and so qualifies as black-box testing.
blackbox testing

In TDD, think of tests written to capture specifications as executable specifications.
whitebox testing

Can anyone describe what this is?
whitebox testing

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

```java
if (x < y) {
    z = f(x, y);
} else {
    z = g(x, y, z);
}
```
whitebox testing

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

```java
if (x < y) {
    z = f(x, y);
} else {
    z = g(x, y, z);
}
```

input → [Code block] → output
whitebox testing

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

```java
if (x < y) {
    z = f(x, y);
} else {
    z = g(x, y, z);
}
```
Code coverage

We will use gcov as our coverage tool.

Compile with,

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

```
gcc $(CFLAGS) -fprofile-arcs -ftest-coverage
-L /util/CUnit/lib
-I /util/CUnit/include/CUnit/
$(OBJECTS) tests.c -o tests
-lcunit -lgcov
```
Instrument arcs during compilation. For each function of your program, GCC creates a program flow graph, then finds a spanning tree for the graph.

https://gcc.gnu.org/onlinedocs/gcc-2.95.2/gcc_2.html#SEC9
Create data files for the gcov code-coverage utility (see section gcov: a GCC Test Coverage Program).

https://gcc.gnu.org/onlinedocs/gcc-2.95.2/gcov_2.html#SEC9
-llibrary

Search the library named library when linking.
It makes a difference where in the command you
write this option; the linker searches/processes
libraries and object files in the order they are
specified. Thus,

    foo.o -lz bar.o

searches library `z' after file `foo.o' but before `bar.o'.
If `bar.o' refers to functions in `z', those functions
may not be loaded.

[...]
The directories searched include several standard
system directories plus any that you specify with `-L'.

https://gcc.gnu.org/onlinedocs/gcc-2.95.2/gcc_2.html#SEC13
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)
Exercise:
https://tools.ietf.org/html/rfc3986#section-3.1

Clone:
https://classroom.github.com/a/YSDeCB4W

Docs:
https://gcc.gnu.org/onlinedocs/gcc/Invoking-Gcov.html#Invoking-Gcov