## Week 5-6

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Testing I</td>
<td>• Group meetings</td>
<td>• Midterm</td>
<td>• No Section</td>
<td>• Testing II</td>
</tr>
<tr>
<td>• No reading</td>
<td></td>
<td></td>
<td></td>
<td>• Progress report due</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Readings out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Testing III</td>
<td>• Group meetings</td>
<td>• TBA</td>
<td>• Section</td>
<td>• ZFR demos</td>
</tr>
<tr>
<td>• Readings due</td>
<td></td>
<td></td>
<td>• ZFR due</td>
<td></td>
</tr>
</tbody>
</table>

White box: slides and a demo
Today: white box testing
(≡ clear box = transparent box = glass box)

- Goals
  - Ensure test suite covers (executes) all of the program
  - Measure quality of test suite with % coverage
- Assumption
  - High coverage → few mistakes in the program
  - Assumes test produce expected output
- Focus: features not described by specification
  - Control-flow details
  - Performance optimizations
White-box motivation

• Some pieces of code are hard to test fully without knowing the code

```java
boolean[] primeTable = new boolean[CACHE_SIZE];

boolean isPrime(int x) {
    if (x > CACHE_SIZE) {
        for (int i = 2; i < x / 2; i++) {
            if (x % i == 0) return false;
        }
        return true;
    } else {
        return primeTable[x];
    }
}
```

• Important transition around $x = CACHE_SIZE$ that would be hard to guess at since `CACHE_SIZE` is hidden from the interface
White Box Testing: Advantages

• Finds an important class of boundaries
  – Yields useful test cases

• Consider \texttt{CACHE\_SIZE} in \texttt{isPrime} example
  – Need to check numbers on each side of \texttt{CACHE\_SIZE}
    • \texttt{CACHE\_SIZE-1}, \texttt{CACHE\_SIZE}, \texttt{CACHE\_SIZE+1}
      – If \texttt{CACHE\_SIZE} is mutable, may need to test with different \texttt{CACHE\_SIZES}

• Disadvantages?
  – Tests may have same bugs as implementation
  – What’s a statement?
What is full coverage?

```c
static int min (int a, int b) {
    int r = a;
    if (a <= b) {
        r = a;
    }
    return r;
}
```

- Consider any test with $a \leq b$ (e.g., $\text{min}(1,2)$)
  - It executes every instruction
  - It misses the bug
- Statement coverage is not enough
Edge coverage

• Another approach is to use a control flow graph (CFG) representation of a program
  – Essentially, a flowchart

• Then ensure that the suite covers all edges in the CFG

```c
static int min (int a, int b) {
    int r = a;
    if (a <= b) {
        r = a;
    }
    return r;
}
```
Condition coverage

- Complex conditions can confound edge coverage
  - if (p != NULL) and
    (p->left < p->right) ...
- Is this a single conditional statement in the CFG?
- How are short-circuit conditionals handled?
  - andthen, orelse
Path coverage

• Edge coverage is in some sense very static
• Edges can be covered without covering paths (sequences of edges)
  – These better model the actual execution
if (x != 0) {
y = 5
} else {
z = z-x
}

if (z > 1) {
z = z/x
} else {
z = 0
}

T1: \{x=1, z=2\}

T2: \{x=0, z=-2\}
Example
Path coverage and loops

- In general, we can’t bound the number of times a loop executes
- So there are an unbounded number of paths in general
- Often in practice
  - Clear boundary conditions
  - 10
Varieties of coverage

- Statement coverage
- Branch coverage
- Decision coverage
- Loop coverage
- Condition/Decision coverage
- Path coverage
Limitations of coverage

- 100% coverage is not always a reasonable target
- 100% may be unattainable (dead code)
- High cost to approach the limit
- Coverage is just a heuristic
- Oracles – “does it do the right thing?” – are often neglected in the face of coverage
- High-coverage can be a counter-intuitive indicator of poor code
How to increase coverage?

- Another limitation is that, beyond simple examples, it is often hard to figure out how to increase coverage – that is, what tests should be added to cover a particular statement or edge or path or such?
- How might you do this?
<table>
<thead>
<tr>
<th>Week 5-6</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>Testing I</td>
<td>Group meetings</td>
<td>Midterm</td>
<td>No Section</td>
<td>Testing II</td>
</tr>
<tr>
<td></td>
<td>No reading</td>
<td></td>
<td></td>
<td></td>
<td>Progress report due</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Readings out</td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
<td>TBA</td>
<td></td>
<td>Section</td>
<td>ZFR due</td>
</tr>
<tr>
<td>Thursday</td>
<td></td>
<td></td>
<td></td>
<td>ZFR due</td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td>Testing III</td>
<td>Group meetings</td>
<td>TBA</td>
<td>Section</td>
<td>ZFR demos</td>
</tr>
<tr>
<td></td>
<td>Readings due</td>
<td></td>
<td></td>
<td>ZFR due</td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
<td>TBA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td></td>
<td>TBA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
<td>TBA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>