INTRODUCTION TO INSPECTION AND TESTING

The game plan

1. Write each of the three programs on the board.

2. For each program, ask:
   - What does it do?
   - Is it correct?

3. Establish:
   - "Correctness" has meaning only with respect to some intended purpose
   - To do "engineering" you must record that intended purpose in a carefully written specification.

Three programs

swap

```
/* specification: swap x and y
   Assume: no arithmetic overflow
*/

x = x + y;
y = x - y;
x = x - y;
```

MORAL: the implementation need not resemble the specification.
Ideally, the specification is simpler.

sum

```
/* specification:
 * if n > 0 then
 *      s = s + sum of [n,n-1,...,1]
 *  else
 *      no change
 */

 OR

 /* if n >= 0 and s = 0 then
 *      s = sum of [n,n-1,...,1]
 */

while (n > 0) {
    s = s + n;
    n--;
}
```

MORAL: sometimes a partial specification is better, for simplicity and to focus on the intended use.

3x+1

```
/* specification:
 * if it terminates: x == 1; but does it terminate?
 */

while (x != 1) {
    if (x % 2 == 0)
        x = x/2;
    else
        x = 3*x + 1;
}
```

MORAL: a short program may have very complex behavior

TERMS

specification	a precise description of the required observable behavior of a program
implementation	source code
verification	showing that code (or documentation) is correct
fault	an error in source code (or documentation)
failure	the occurrence of incorrect program behavior
inspection	human review of source code with the goal of revealing faults
testing	running a program on inputs chosen to reveal failures

Next

The inspection method ...