07-120 Introduction to Software Construction

Fall 2023

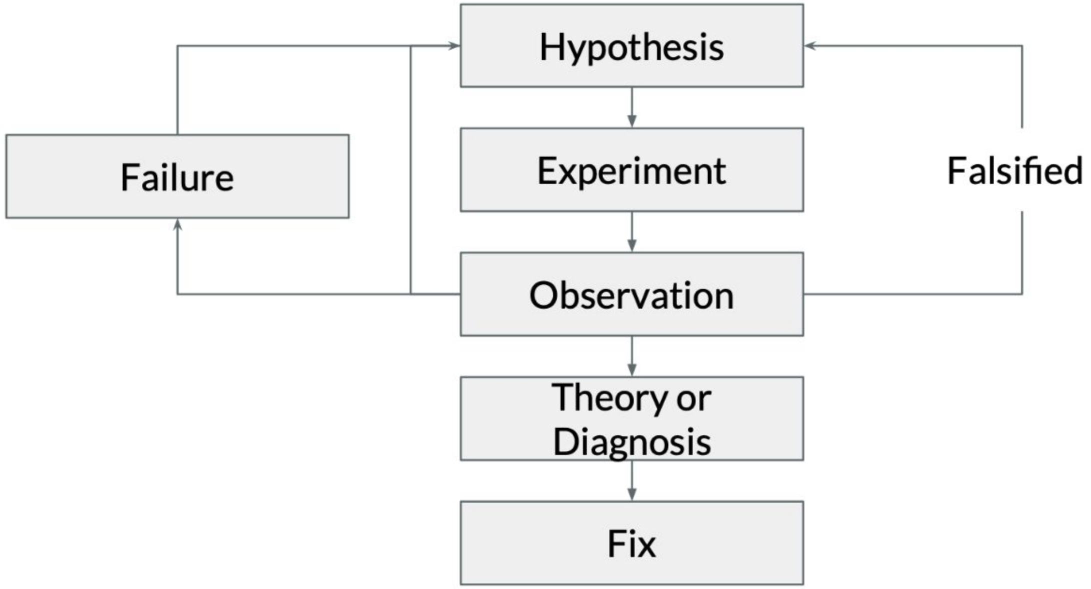
Michael Hilton and Mayank Goel

Administrivia

HW3 Due Date: Tuesday, November 7, 2023 at 11:59 pm.

Debugging

Scientific Debugging



Document your process

Hypothesis	
Prediction	
Experiment	
Observation	
Conclusion	

OBSERVATION:
Continually having to re-observe past behavior takes time.

Testing

Testing

Testing can only show the presence of errors – not their absence. (Dijkstra 1972)

What are the goals of testing?

What are the goals of testing?

Reveal Failures

Access quality

Clarify the Specification

Learn about the program

Document the behavior

How do you feel about testing?

What sort of values should we test?

Suppose we were going to test a function that

The ideal test suite

Uncovers all errors in code that are detectable through testing

Uncovers all errors in requirements capture

All scenarios covered

Is of minimum size and complexity

Uncovers errors early in the process, ideally when code is being written (“test cases first”)

Testing in practice

Test suites are a tradeoff between completeness, and time possible.

Contracts

Contracts

A single observation can help us debug

Tests can re-run a single observation over and over

Contracts can ensure a range of observations are correct.

Review: possible contracts in co

@requires

@ensures

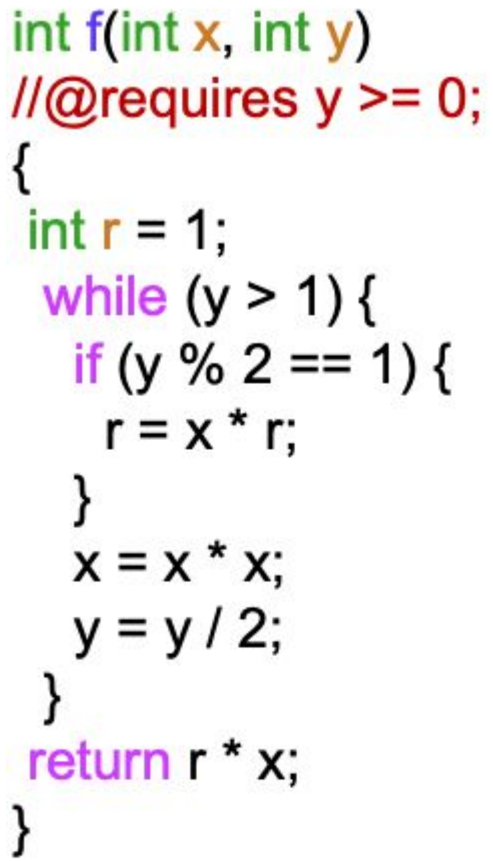
@loop_invariant

@assert

NOTE: Purity enforcement

@requires

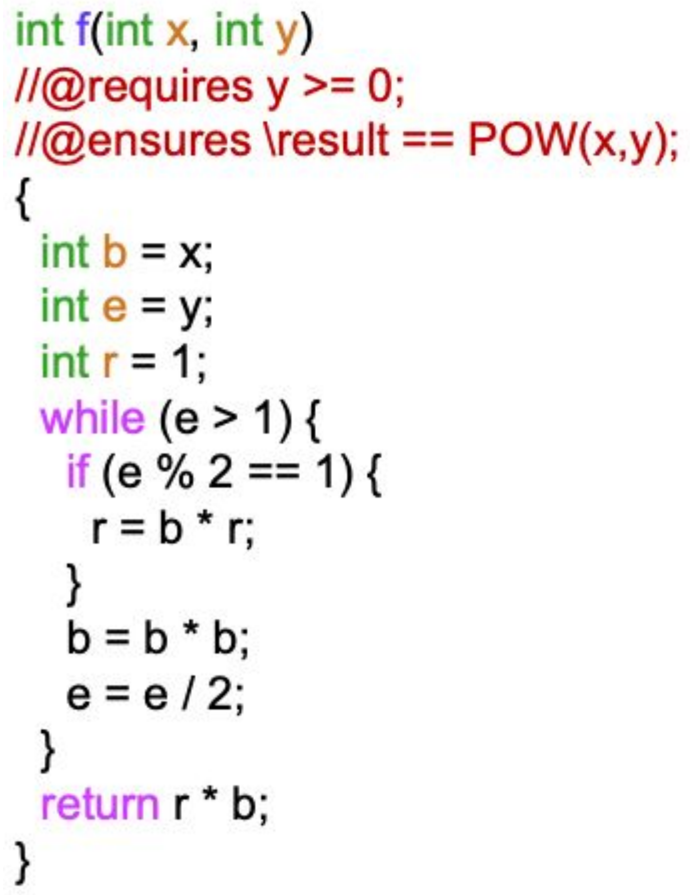
//@requires y >= 0;



```
int f(int x, int y)
//@requires y >= 0;
{
 int r = 1;
  while (y > 1) {
    if (y % 2 == 1) {
      r = x * r;
    }
    x = x * x;
    y = y / 2;
  }
 return r * x;
}
```

@ensures

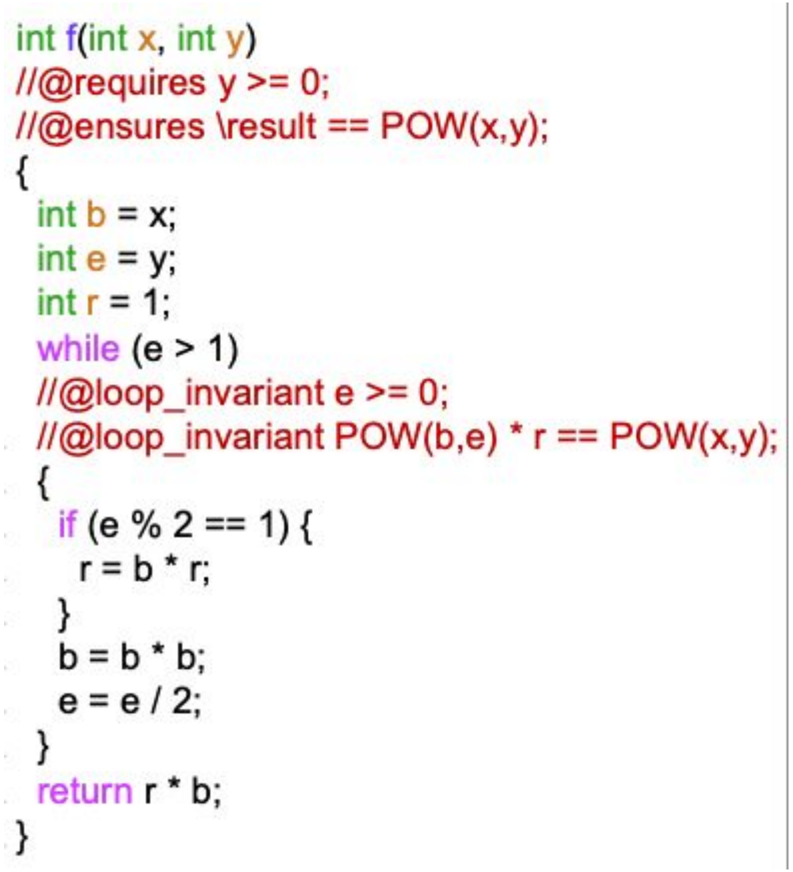
//@ensures \result == POW(x,y) ;



```
int f(int x, int y)
//@requires y >= 0;
//@ensures \result == POW(x,y);
{
  int b = x;
  int e = y;
  int r = 1;
  while (e > 1) {
    if (e % 2 == 1) {
      r = b * r;
    }
    b = b * b;
    e = e / 2;
  }
  return r * b;
}
```

@loop

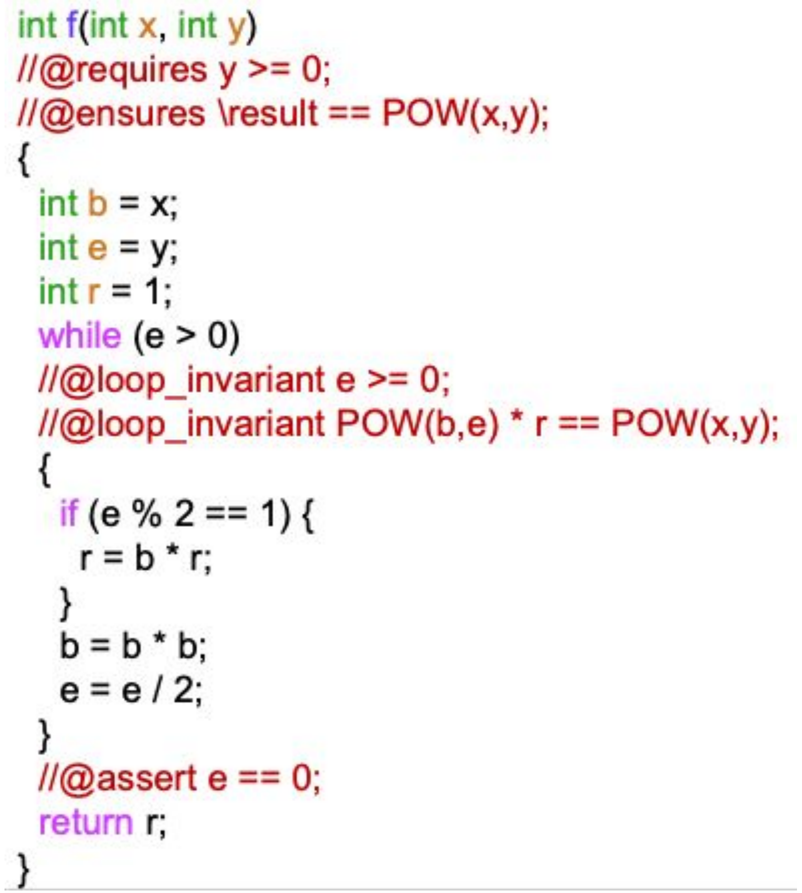
//@loop_invariant e >= 0;



```
int f(int x, int y)
//@requires y >= 0;
//@ensures \result == POW(x,y);
{
  int b = x;
  int e = y;
  int r = 1;
  while (e > 1)
  //@loop_invariant e >= 0;
  //@loop_invariant POW(b,e) * r == POW(x,y);
  {
    if (e % 2 == 1) {
      r = b * r;
    }
    b = b * b;
    e = e / 2;
  }
  return r * b;
}
```

@assert

//@assert e == 0;



```
int f(int x, int y)
//@requires y >= 0;
//@ensures \result == POW(x,y);
{
  int b = x;
  int e = y;
  int r = 1;
  while (e > 0)
  //@loop_invariant e >= 0;
  //@loop_invariant POW(b,e) * r == POW(x,y);
  {
    if (e % 2 == 1) {
      r = b * r;
    }
    b = b * b;
    e = e / 2;
  }
  //@assert e == 0;
  return r;
}
```

Tests vs Contracts

Suppose we want to test a function

$y < x+2$

Tests for $y < x+2$

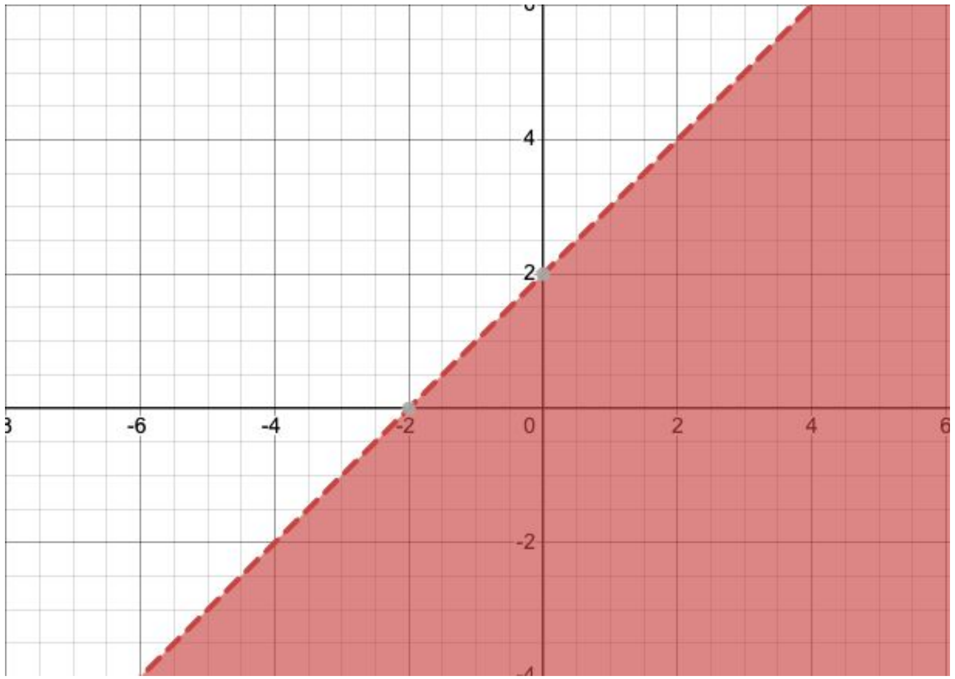
Specific observations that can tell us if we are correct or not.

y	x	True?
1	2	Y
10	4	F
…	…	…

Contracts for y < x+2

How can we test ALL values?

@ensures \result == y < x+2 ;



Exercise - Blackjack (aka 21)

- Standard US deck
    - 4 suits (clubs, hearts, diamonds, spades)
    - Ace, 2-10,J,KQ,K
    - 52 cards total
- Dealer deals 2 cards to player, and 2 to themselves.
    - 1 face up, 1 face down
    - All cards count as face value, Ace is 1 or 11, pictures count as 10. Suits don't matter
- Player can stand (keep cards), hit (get a new card), surrender(fold), double down (double your bet), or split (start two new hands, one with each card from old hand)
- Dealer must hit on =<16 and stand on >=17
- Bust is going over 21 for hand
- Player wins when they hand total exceeds the dealer, or if dealer bust. Payoff is 1 to 1
- Players lose if they bust
- If tie, money is kept by player

Activity

Write contracts on paper

Talk with another student

Write contracts on board

Discuss using contracts to design your program