Outline

Correctness

Correctness of Iterative Algorithms

Correctness of Recursive Algorithms
Correctness

- How do we know that our algorithms are correct?

- What does it mean to say an algorithm is “correct”? 
Correctness
Some definitions

- Preconditions

- Postconditions
Correctness

Some definitions

- Termination

- Partial correctness
def power(x, y):
    z = 1
    m = 0

    while m < y:
        z = z * x
        m = m + 1

    return z
Integer Power
Algorithm to compute $x^y$

```
# Pre: x in R, y in N
def power(x, y):
    z = 1
    m = 0
    # loop pre: x in R, y in N, z = 1, m = 0
    while m < y:
        # LI(m, z) : 0 <= m <= y and z = x^m
        z = z * x
        m = m + 1
    # loop post: z = x^y
    return z
# Post: returns x^y
```
Correctness by Design
Consider before, during, after
Iterative Binary Search

# Pre: A is sorted and x comparable with A[0..n-1]
def binSearch(x, y):
    b = 0
    e = n - 1
    # LI: 0 <= b <= e+1 <= n and A[0..b] <= x <= A[e+1..n]
    while b < e:
        m = (b + e) // 2
        if A[m] < x:
            b = m + 1
        else:
            e = m
    # Post: 0 <= b <= n and A[0..b-1] < x <= A[b..n-1]
    if A[b] == x:
        return b
    else:
        return -1
Termination?
def recBinSearch(x, A, b, e):
    if b == e:
        if A[b] == x:
            return b
        else:
            return -1
    else:
        m = (b + e) // 2  # midpoint
        if x <= A[m]:
            return recBinSearch(x, A, b, m)
        else:
            return recBinSearch(x, A, m+1, e)
Preconditions and Postconditions for RecBinSearch
Precondition $\Rightarrow$ Termination and Postcondition

Proof: induction on $n = e - b$

Case 1: $x \leq A[m]$

Case 2: $x > A[m]$