Review 5

Recursion
What We Will Do today

• Practice recursive specifications and functions
  ▪ Given a recursive problem definition
    • Determine a proper specification (note preconditions)

  ▪ Given a problem description and specification:
    • Write the recursive base case
    • Write the recursive call
    • Verify that it is correct

Questions?
Important Steps

1. Precise Specification
   - What does the function do?
   - What are the preconditions?

2. Write the base case
   - What is the most basic case?
   - What causes termination of the recursive function?

3. Write the recursive case
   - How do we make progress toward termination?
   - Is your computation correct?
Write a specification for a function that:

- Computes the complement of a positive integer. i.e. The complement of 12345 is 98765.

- Reduce the positive input integer to a single digit. i.e. 472 \rightarrow 47 + 2 = 49 \rightarrow 4 + 9 = 13 \rightarrow 1 + 3 = 4
Write a specification for a function that:

- Computes the complement of a positive integer. i.e. The complement of 12345 is 98765.

  """Returns: complement of n, by replacing each decimal digit of n by 10-n. i.e. the result for 93723 is 17387. Precondition: n > 0 an int, and no digit of n is 0""

- Reduce the positive input integer to a single digit. i.e. 472 -> 47+2 = 49 -> 4+9 = 13 -> 1+3 = 4

  """Returns: n reduced to a single digit (summing its digits) Precondition: n > 0 an int"""
Writing Specifications

Write a specification for a function that:

- Compresses a String so that duplicate are replaced with counts i.e. aaabbbbbbbccd -> a3b6c2d1

- Converts an integer to a string representation with commas i.e. 5923821 is converted to 5,923,821.
Writing Specifications

Write a specification for a function that:

- Compresses a String so that duplicate are replaced with counts
  i.e. aaabbbbbbcdd -> a3b6c2d1

  """Returns: s compressed so that duplicates are replaced with count of how many occurrences that character has in a row. Precondition: s a string""

- Converts an integer to a string representation with commas
  i.e. 5923821 is converted to 5,923,821.

  """Returns: String representation of n with commas added Precondition: n an int (positive or negative)"""
def complement(int n) {
    """Returns: the complement of n, formed by replacing each decimal digit of n by 10-n.
    i.e. the result for the integer 93723 is 17387.
    Precondition: n > 0 and int, and no digit of n is 0"
    # Base Case

    # Recursive Case
def complement(int n) {
    "Returns: the complement of n, formed by replacing each decimal digit of n by 10-n.
    i.e. the result for the integer 93723 is 17387.
    Precondition: n > 0 and int, and no digit of n is 0"
    # Base Case
    if n < 10:
        return 10 - n
    # Recursive Case
    return complement(n/10) * 10 + (10 - n%10)
def add_commas(n):
    
    """Returns: string representation of n with commas added
    Precondition: n is an int (positive or negative)"""

    # Base case

    # Recursive Case
def add_commas(n):
    """Returns: string representation of n with commas added
    Precondition: n is an int (positive or negative)""
    # Base case
    if n < 1000:
        return str(n)
    # Recursive Case
    number = str(n)
    return add_commas(n/1000) + ',' + number[-3:0]
def add_commas(n):
    """Returns: n with commas added. Precondition: n is an int (positive or negative)""
    if n < 0:
        return '-' + add_commas_helper(-n)
    else:
        return add_commas_helper(n)

def add_commas_helper(n):
    """Returns: n with commas added. Precondition: n > 0 is an int""
    # Base case
    if n < 1000:
        return str(n)
    # Recursive Case
    number = str(n)
    return add_commas_helper(n/1000) + ',' + number[-3:]
Recursion and Objects

- Class Person (person.py)
  - Objects have 3 attributes
    - name: String
    - mom: Person (or None)
    - dad: Person (or None)
  - Represents the "family tree"
    - Goes as far back as known
    - Attributes mom and dad are None if not known
- Constructor: Person(n,m,d)
  - Or Person(n) if no mom, dad
def num_ancestors(p):
    """Returns: num of known ancestors
    Pre: p is a Person"
    # Base case
    # No mom or dad (no ancestors)
    # Recursive step
    # Has mom or dad
    # Count ancestors of each one
    # (plus mom, dad themselves)
    # Add them together
def num_ancestors(p):
    """Returns: num of known ancestors
    Pre: p is a Person"""
    # Base case
    if p.mom == None and p.dad == None:
        return 0
    # Recursive step
    moms = 0
    if not p.mom == None:
        moms = 1+num_ancestors(p.mom)
    dads = 0
    if not p.dad == None:
        dads = 1+num_ancestors(p.dad)
    return moms+dads

John Sr.  Pamela  ???
          |       |
          John Jr. Jane Dan  Heather
          |       |
          John III Jane Robert  Ellen
          |       |
          John IV Alice  ???

11 ancestors
Extra Problems

• Given a list, use recursion to determine if it is sorted

• Given a String $s$, list all the permutations of String $s$:
  - “XZY” $\rightarrow$ “XYZ”, “XZY”, “ZYX”, “YXZ”, etc

• Use recursion to find the minimum element in a list
class FacebookProfile(object):
    """name [str]: name of this profile
    friends [list of FacebookProfile]: friends list"""

We want to answer the question:
• Is this profile at most 6 degrees away from Kevin Bacon?
• In other words, is Kevin Bacon a friend of a friend of a friend of a friend of a friend of a friend?

Specification (Method inside class FacebookProfile):
    def sixDegreesOfBacon(self):
        """Returns: True if this FacebookProfile is at most 6 degrees away from Kevin Bacon; False otherwise"""
class FacebookProfile(object):
    ...

def sixDegreesOfBacon(self):
    """Returns: True if this FacebookProfile is at most 6 degrees away from Kevin Bacon"""
    return self.sixDegreesHelper(6)

def sixDegreesHelper(self, n):
    """Returns: True if this FacebookProfile is at most n degrees away from Kevin Bacon
    Precondition: n > 0 an int"""
    # Base case
    if self.name == 'Kevin Bacon':
        return True
    if n == 0:
        return False
    # Recursive Case
    for f in self.friends:
        if f.sixDegreesHelper(n-1):
            return True
    return False
Questions?