Exam Info

- Prelim 2: 7:30–9:00PM, Tuesday, April 25th
  - aa200 – jjm200  Baker Laboratory 200
  - jjm201 – sge200  Rockefeller 201
  - sge201 – zz200  Rockefeller 203
- Baker Lab 200, Rockefeller Hall 201, 203
- No Electronics, No Notes, Closed book.
- Bring your Cornell ID
- Put your Name & NetId on Each Page!!!
What is on the Exam?

• The big topics:
  ▪ Nested Lists and Dictionaries (Lab 8, A3)
  ▪ Recursion (Lab 9, A4)
  ▪ Classes and subclasses (Lab 10, Lab 11, A4)
  ▪ While loops
What is on the Exam?

- The big topics:
  - Nested Lists and Dictionaries (Lab 8, A3)
  - Recursion (Lab 9, A4)
  - Classes and subclasses (Lab 10, Lab 11, A4)
  - While loops
Diagram the objects created during the following code:

```python
>>> nlst = [[1, 2], [3, 4, 5], [6, 7]]
>>> slice = nlst[1:]
>>> slice[1].append(0)
```
def max_cols(table):
    ""
    Returns: Row with max value of each column
    
    We assume that table is a 2D list of floats (so it is a list of rows and each row has the same number of columns. This function returns a new list that stores the maximum value of each column.
    
    Examples:
    
    max_cols([ [1,2,3], [2,0,4], [0,5,2] ]) is [2,5,4]
    max_cols([ [1,2,3] ]) is [1,2,3]
    
    Precondition: table is a NONEMPTY 2D list of floats"""
Dictionaries

• Key-value pairs, unique keys
• Creation: dic = {'a': 1, 'b': 2, 'c': 3}
• Access: dic['a']
• Modification: dic['a'] = 5
• Add new key: dic['d'] = 7
• Does not have a specific order! Not indexable
What is on the Exam?

• The big topics:
  ▪ Nested Lists and Dictionaries (Lab 8, A3)
  ▪ Recursion (Lab 9, A4)
  ▪ Classes and subclasses (Lab 10, Lab 11, A4)
  ▪ While loops
Recursion

• What kind of questions might be asked?
  ▪ Will be given a function specification
  ▪ Implement it using recursion
  ▪ May have an associated call stack question

• Divide and Conquer
  ▪ Base case
    • Decide what to do on “small” data
  ▪ Recursive case
    • Decide how to break up your data into smaller pieces
  ▪ Decide how to combine your answers
def flatten(lst):
    """Return: a COPY of the flattened version of the list lst.

    lst is a potentially nested list. A flattened version of lst means to take the nested list and turn it into a one-dimentional list.

    Example: flatten([]) returns [],
    flatten([[1, 2, 3]]) returns [1, 2, 3]
    flatten([1, [2, 3], [[4], []], 5, [6, 7, 8]], 9)) returns [1, 2, 3, 4, 5, 6, 7, 8, 9]

    Precondition: lst is a list or an int"""
class Person(object):

    '''Instance is a person/family tree
    INSTANCE ATTRIBUTES:
    name: First name [nonempty str]
    mom: Mom’s side [Person or None]
    dad: Dad’s side [Person or None]
    '''

    ...

To make person s in the right picture, you do s = Person(‘Jane’, None, None)
To make person q, you use the assignment q = Person(‘Robin’, s, None)

A genealogy list is defined recursively as follows:

- It is a nonempty list with exactly three elements.
- The first element is a nonempty string, representing the person’s name.
- The last two elements are either None or genealogy lists.

For example, the genealogy list of s is [‘Jane’, None, None]
The genealogy list of q is [‘Robin’, [‘Jane’, None, None], None]
def geneology_list(person):
    """Return: A geneology list of the Person object, person.

    For example, using the objects on the previous slide,
geneology_list(s) returns ['Jane', None, None]
geneology_list(q) returns ['Robin', ['Jane', None, None], None]

    Precondition: person is a Person object or None
    """""""
def histogram(s):
    """Return: a histogram (dictionary) of the # of letters in string s.
    The letters in s are keys, and the count of each letter is the value. If
    the letter is not in s, then there is NO KEY for it in the histogram.

    Example: histogram("") returns {},
    histogram('abracadabra') returns {'a': 5, 'b': 2, 'c': 1, 'd': 1, 'r': 2}

    Precondition: s is a string (possibly empty) of just letters."""

def histogram(s):
    """Return: a histogram (dictionary) of the # of letters in string s.

    The letters in s are keys, and the count of each letter is the value. If the letter is not in s, then there is NO KEY for it in the histogram.

    Precondition: s is a string (possibly empty) of just letters.""

**Hint:**

- Use divide-and-conquer to break up the string
- Get two dictionaries back when you do
- Pick one and insert the results of the other
Recursion and the call stack

```python
def skip(s):
    """Returns: copy of s
    Odd (from end) skipped"
    result = ''
    if (len(s) % 2 == 1):
        result = skip(s[1:])
    elif len(s) > 0:
        result = s[0]+skip(s[1:])
    return result
```

- **Call**: `skip('abc')`
- Recursive call results in four frames (why?)
  - Consider when 4th frame completes line 6
  - Draw the entire call stack at that time
- Do not draw more than four frames!
Any More Questions?