CS1110 Lecture 12: Intro to Recursion

Upcoming schedule

Today: Return of graded A2s to study from.
Deadline for final A1 submission (unless you got an extension beyond that)

Tuesday Mar 11
- Lecture = (optional) review session; can also pick up graded A2
- Lab sessions = (optional) drop-in office hours; can also pick up graded A2
- Prelim: 7:30-9pm, 200 Baker Lab (= the auditorium)

Wednesday Mar 12: No labs or office hours

Prelim grades announced by email from CMS, hopefully by Friday morning.

Linguistic chunks as nested lists

- ['hit', 'a', 'guy', 'with', 'glasses']
  - modifies what?

- ['hit', ['a', ['guy', ['with', 'glasses']]]]
  - in 4 lists

- ['hit', ['a', 'guy'], ['with', 'glasses']]
  - in 2 lists

Let's take "list embeddedness" as indication of structure:
what's the max number of lists within which an object is enclosed?

How might you implement this?

```python
def embed(input):
    '''Returns: depth of embedding in input.
    Precondition: input is a list of strings or a string'''
    if type(input) != list:  # base case
        return 0
    else:
        return 1 + max(map(embed, input))
```

Test cases

```python
def embed(input):
    '''Returns: depth of embedding in input.
    Precondition: input is a list of strings or a string'''
    if type(input) != list:  # base case
        return 0
    else:
        the function embed uses itself
        return 1 + max(map(embed, input))
```

Preparing for the exam

Past exams are posted on the Exams section of the website.
Profs Lee and Marschner wrote the Spring 2013 one, but that exam didn't cover map or for-loops.

Recommended preparation: Review all lectures up to and including March 4. Be able to do A1, A2, labs 1-5 from scratch, cold. For lab 6, be comfortable with writing for-loops and employing map in the ways the lab asks you to.

As always, come to our many office hours/consulting hours for in-person help; see the Staff section of the webpage.

As always, watch Piazza for announcements, for helpful answers to other people's questions, etc.

Preparing for the exam

Test cases

```python
def embed(input):
    '''Returns: depth of embedding in input.
    Precondition: input is a list of strings or a string'''
    if type(input) != list:  # base case
        return 0
    else:
        if type(input) == list:
            # map
            item_embed = map(embed, input)
            return max(item_embed) + 1
        else:
            return 1
```

One solution

```python
def embed(input):
    '''Returns: depth of embedding in input.
    Precondition: input is a list of strings or a string'''
    if type(input) != list:  # base case
        return 0
    else:
        if type(input) == list:
            # map
            item_embed = map(embed, input)
            return max(item_embed) + 1
        else:
            return 1
```
How to Think About Recursive Functions

- Have a precise function specification.
- Figure out how to handle the base case(s):
  Base cases: argument values are as “small” as possible, or when the answer is determined with little calculation.
- Figure out how to handle the recursive case(s):
  - How can the problem be described as the combination of answers to smaller versions of the original?
- Figure out how to perform the decomposition:
  - Arguments of calls must somehow get "smaller", so each recursive call gets closer to a base case.

Simpler, restricted example

```python
def num_es(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    pass

Recursive idea: If s has at least one character, the number of 'e's in s is the number of 'e's in s[0] + the number of 'e's in s[1:]
'a': 0 = 0 + 0
'cag': 0 = 1 + 0
'e': 0 = 1 + 0 # trick using s[1:] == ""
'ceddd': 2 = 0 + 2
```

Simpler example:
Recursive, restricted version of count

```python
if s has at least one character, the number of 'e's in s is the number of 'e's in s[0] + the number of 'e's in s[1:]
```

```python
def num_es(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    return (startcount + num_es(s[1:]))
    # s[1:] is "" if len(s) == 1
```

Base case part of the code

```python
def num_es(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    # case: s is empty string
    if s == '':
        return 0
        # Base case
    else:
        # case: s has at least one char
        return 1 if s[0] == 'e' else 0 + num_es(s[1:]),
```

Recursive part of code

```python
def num_es(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    if s[0] == 'e':
        startcount = 1
    else:
        startcount = 0
    return (startcount + num_es(s[1:]))
    # s[1:] is "" if len(s) == 1
```

Example: Remove Blanks from a String

```python
def deblank(s):
    """Returns: s with blanks removed""
    if s == '':
        return s
    if s[0] in string.whitespace:
        return deblank(s[1:]),
    return (s[0] + deblank(s[1:]))
```

- Check the four points:
  1. Precise specification?
  2. Base case: correct?
  3. Recursive case: progress toward termination?
  4. Recursive case: correct?