CS 97 Discussion Section

Lun Liu, 1D

Welcome back to CS97 Discussion!

- HW #2 is due Monday night.
- HW #1 is still being graded.
- Full solutions to practice problems will be posted on CCLE over the weekend.
- Midterm #1 is on Wednesday
- Please take this survey for our learning assistants: http://bit.ly/2ipoQfr

Common Mistakes-Week 3 Edition:

.There are no bad ideas, only good ideas that go horribly wrong

- Examples are a double-edged sword
  - It's nice to look at references, but for every line of code, try to engage with it instead of just copying it exactly. Why is that line there? What does it do? What would happen if you removed it?
- Mismatched operand types.
  - Remember that things like +, -, /, >, <, == are operators, and they need two of things the same type (list, string, number, bool)
- Not checking all of your return statements!
  - Since we're writing a lot of recursive functions, there will usually be more 2+ return statements. If your function has an error it's tempting to only check the last, but check them all.
- Don't forget the small stuff.
  - Now that you're writing fancy recursive functions, if you really can't figure out an error, sometimes it's as simple as missing ':'!

Good God, Lemon!

// This function is supposed to return the sum of all of the even numbers in a list, but it has a lot of bugs. Find them! Go to collabedit.com/yx5wq to get a copy-paste version of this program to put into IDLE.

```
def sumEvens(lst):
        if lst = []:
                return []
        elif lst[0] % 2 == 0
                return l[0] + sumEvens(lst)
        else lst[0] % 2 != 0:
                return sumEven(lst)
```

Calm down, take a deep breath, and prepare for the Thunderdome

Remember the "pow" function? It takes two numbers, n and m and returns n^m. (In Python, you can also use the ** operator to achieve this effect!) Implement the pow function; your version is dumber than the library version and won't work if m is not a nonnegative integer.

Recursion Review: Strategies

If you're stuck, try answering the following questions for this problem:

How can I break this into smaller pieces? What's the smallest possible input? How do I make it bigger or smaller? (recursive strategy)

When are some situations I have enough information to give a definite answer to this question?   (base cases)

What *operations* do I have to do repeatedly to answer this question? (recursive body)

Calm down, take a deep breath, and prepare for the Thunderdome

Remember the "pow" function? It takes two numbers, n and m and returns n^m. (In Python, you can also use the ** operator to achieve this effect!) Implement the pow function; your version is dumber than the library version and won't work if m is not a nonnegative integer.

```
def pow(n, m):
        if m == 0:
                return 1
        return n * pow(n, m-1)
```

I'm Scared, But It Tickles

```
def mystery(str1, str2):
        if len(str1) != len(str2) or len(str1) < 1:
                return ""
        if len(str1) == 1:
                return str1[0] + str2[0]
        return str1[0] + str2[0] + mystery(str1[1:], str2[1:])

>> mystery("sok", "poy")
//How many times is it called? With what? Final output? General
description?
```

Auxiliary/Helper Functions:

Sometimes, recursive problems can only be solved (or can be solved more easily) by using an extra function that does all the extra work.

```
def index(l):
  """
    Convert a list into an indexed list:

    >>>   index(["i", "a", "m", "a"])
    [["i", 0], ["a", 1], ["m", 2], ["a", 3]]
    """
```

Index: A Possible Solution

```
def idx_helper(l, n):
    if len(l) == 0:
        return []
    else:
        head = l[0]
        tail = l[1:]
        return [head, n] + idx_helper(tail, n + 1)

def index(l):
    return idx_helper(l, 0)
```

Write a function called zip...

```
def zip(l1, l2):
    """
    l1, l2 are two lists of same length
    Returns a zipped list
    >>> zip([1,2,3], ["one", "two", "three"])
    [[1,"one"], [2, "two"], [3, "three"]]
    >>> zip([], [])
    []
    >>> zip([2], [22])
    [[2, 22]]
    """
```

Solutions: zip

```
def zip(l1, l2):
    if len(l1) == 0:
        return []
    else:
        head1 = l1[0]
        tail1 = l1[1:]
        head2 = l2[0]
        tail2 = l2[1:]
        new_head = [head1, head2]
        new_tail = zip(tail1, tail2)
        return [new_head] + new_tail
```

...and one called unzip.

```
def unzip(l):
    """
    l is zipped list
    Returns a list of two unzipped lists

    >>> unzip([[1,"one"], [2, "two"], [3, "three"]])
    [[1,2,3], ["one", "two", "three"]]
    >>> unzip([])
    [[], []]
    >>> unzip([[1, 11]])
    [[1], [11]]
    """
```

Solutions: unzip

```
def unzip(l):
    if len(l) == 0:
        return [[], []]
    else:
        head = l[0]
        tail = l[1:]
        l1_head = head[0]
        l2_head = head[1]
        unzipped_tails = unzip(tail)
        l1_tail = unzipped_tails[0]
        l2_tail = unzipped_tails[1]
        return [[l1_head] + l1_tail, [l2_head] + l2_tail]
```

countDigits

```
def countDigits(n):
    """
    return number digits in a number. Assume n is a positive
     integer. Hint: play with / and % in IDLE!"""
    >>> countDigits(9)
    1
    >>> countDigits(12345)
    5
    >>> countDigits(320190)
    6
    """
```

Solutions: countDigits

```
def countDigits(n):
    if n < 10:
        return 1
    else:
        lastDigit = n % 10
        rightShiftOne = (n - lastDigit) / 10
        return 1 + countDigits(rightShiftOne)
```

Good Luck!