

# CS 97 Discussion Section

---


Lun Liu, 1D

# Common Mistakes: Midterm Edition

- Notable difficulties
  - Q1: variables can be reassigned; if followed by if is not mutually exclusive
  - Q2: recursion isn't always necessary! you've learned to use a hammer, but not everything is a nail
  - Q3: execution order --> build up from substitution
- Overall, good job!

# Midterm Question 3-Execution Order


```
def mystery(lst, n):  
    if n == 0:  
        return lst  
    elif lst == []:  
        return lst  
    else:  
        return mystery(lst[1:], n-1) + [lst[0]]
```



**First (initial) call:** `mystery([1, 3, 5, 7, 9], 2)`  
    `return mystery([3, 5, 7, 9], 1) +`  
`[1]`

**Second (recursive) call:** `mystery([3, 5, 7, 9], 1)`  
    `return mystery([5, 7, 9], 0) +`  
`[3]`

**Third (recursive) call:** `mystery([5, 7, 9], 0)`  
    `return [5, 7, 9]`



`[5, 7, 9, 3, 1]`  
`[5, 7, 9, 3] + [1]`  
`[5, 7, 9] + [3]`  
`[5, 7, 9]`

# Mini-Lesson: Integer Division

- Integer division truncates the decimal part of a division out from the answer
- All languages have some concept of integer versus real number division but have different syntaxes
  - In many languages, integer division is the default!
- The '/' operator is integer division:

```
>> 7 / 2
```

```
3.5
```

```
>> 7 // 2
```

```
3
```

## Why would I tell you this now?

# Unzip Revisited

There's a lot of parallels between this problem and the partition problem from the homework.

```
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]]  
    """
```

# Unzip Revisited

- Unzip

- Input: A single list that contains possibly nothing but maybe **two item lists**:
- Output: A list that contains exactly two lists
- Goal: separate any two item lists with all index 0 items in the first list and all index 1 items in the second list
- Need to recurse over list
- **always put one item in each result list**

- Partition

- Input: A single list that contains possibly nothing but maybe **numbers [**
- Output: A list that contains exactly two lists
- Goal: separate **numbers that match a certain value into list one and the rest in list two**
- Need to recurse over list
- **sometimes place item in first list, sometimes place item in second list**

# Unzip Revisited

```
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]
```

# Sorting Algorithms

Selection sort: Find the person in your group with the minimum number and place them at the sorted group. Find the next person in the unsorted group with the new minimum and place them. Repeat until sorted.

Merge sort: Repeatedly split your group into half until you are in your own group (your group is sorted!). Repeatedly "merge" with other sorted group of your size until sorted.

[https://www.youtube.com/watch?v=XaqR3G\\_NVoo](https://www.youtube.com/watch?v=XaqR3G_NVoo)



# Interesting Links for Sort

- <https://www.toptal.com/developers/sorting-algorithms>