Announcements & Logistics

A1 due Friday Oct 19.
Midterm Tuesday Oct 23.

No prep for next week. (Use the extra time for studying!)
From last time, `LinkedList.__contains__`

We care about running time as a function of input size:

- “constant” \( O(1) \)
- “linear” \( O(n) \)
- “quadratic” \( O(n^2) \)
From last time, `LinkedList.__contains__`

def __contains__(self, item: Any) -> bool:
    curr = self._first
    while curr is not None:
        if curr.item == item:
            return True
    return False
From last time, `LinkedList.__contains__`

Running time can vary, even for a fixed input size!

We’ll revisit this idea later in the course.
Recursion

CSC148, INTRODUCTION TO COMPUTER SCIENCE
DAVID LIU
Data structure informs code structure

```python
i = 0
while i < len(lst):
    ... lst[i] ...
    i += 1

curr = lst._first
while curr is not None:
    ... curr.item ...
    curr = curr.next
```
Data structure informs code structure

```
List[int]

for x in lst:
    ... x ...

List[
    List[int]
]

for lst in lst_of_lsts:
    for x in lst:
        ... x ...
```
Data structure informs code structure

```
List[
    List[
        List[int]
    ]
]

for lst_of_lsts in lst_of_lsts_of_lsts:
    for lst in lst_of_lsts:
        for x in lst:
            ... x ...
```
Data structure informs code structure

A nested list is...

- An integer
- A list of nested lists

```python
def nested_f(obj):
    if isinstance(obj, int):
        ...
    else:
        for sublist in obj:
            ... nested_f(sublist) ...
```
Attempting to fully trace recursive code is time-consuming and error prone.

*When tracing recursive code, don’t trace into recursive calls! Instead, assume each call is correct, and make sure the rest of the code uses those calls correctly.*
Midterm study tips

1. **Concept mastery**: definitions, examples, connections
2. **Code mastery**: make a plan, break down tasks into small chunks, use code templates
3. **Practice in a test-like environment**: pencil and paper, closed book (+ aid sheet), timed environment
Midterm study tips

“Mastery” means that **you** are the expert.

Go beyond “re-reading and re-doing”:
- create multiple solutions
- identify common mistakes or errors, and
- explain ideas to others
- make up new questions
first_at_depth – base case

A single integer is always at depth 0.

```python
>>> first_at_depth(100, 0)
100
>>> first_at_depth(100, 3) is None
True
```
## first_at_depth – recursive case

```python
>>> first_at_depth([10, [[20]], [30, 40]], 2)
30
```

<table>
<thead>
<tr>
<th>sublist</th>
<th>depth</th>
<th>first_at_depth(sublist, depth)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>[[20]]</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>[30, 40]</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>[50]</td>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>
**first_at_depth** – multiple base cases!

```python
first_at_depth(obj, d)
    -> first_at_depth(sublist, d - 1)
```

We are actually recursing on both `obj` and `d`.

Can’t recurse when:
- `isinstance(obj, int)`
- `d == 0`
Nested list **mutation**

Last worksheet on nested lists!

Also really good review for a classic memory-related error.