CS 445

Rec 5

Agenda

1. Housekeeping
2. Topics of This Week
3. Working Session: Lab 4
    a. Goal: insertAtTail() implemented successfully
    b. :-)

Housekeeping

1. **Office hours are in-person only now**
    a. My proposal:
        i. Mon/Wed 9am - 10:30am (in-person)
        ii. 1:1 session (in-person/zoom) by appt
    b. Thoughts? How can I better serve you?
2. **Lab 4 & Project 4 are both up**

Topics of This Week

1. **Recursion**
    a. Definition & implementation
2. **Copy constructor**
    a. Shallow copy vs. deep copy
3. **Purpose of private helper methods in class definition**
    a. To allow direct access to and manipulation of private data

Recursion: In-Class Example

```
public int size()
{
    return sizeHelper(head);
}

private int sizeHelper(Node<T> curr)
{
    if (curr==null)
        return 0;
    return 1 + sizeHelper(curr.next);
}
```

- Always consider the BASE CASE first when writing recursion (Special cases? When do I want the recursion to stop?)
  - Important: don't forget to put a RETURN statement in your base case!
- Think about the action you wish to perform for just ONE Node
- When calling the method ITSELF remember to pass in the next Node for its argument

Lab 4

- **LL_Recursive()**
- **insertAtTail()**
  - We will do a detailed walk-through
- **size()**
  - Explanation and illustration in slow mode
- **toString()**
- **search()**
- **contains()**
  - One liner using search(), no recursion needed, no helper method needed

For each `public` method, write a recursive version with a `private` helper method to allow access to `head` or `curr` based on your iterative solution from Lab 3

insertAtTail()

BASE CASE:

If list is empty (head == `null`)
insertAtFront(head);
return;

OTHERWISE:

“Is `curr` the last Node?”
YES -> insert a new Node and make it the new tail, and return
NO -> insertAtTail() calling the method recursively on the next Node