Tabs in your browser...

A linked list is helpful here!!
...why?
CS 2230
CS II: Data structures

Meeting 6: linked lists
Brandon Myers
University of Iowa
Today’s big ideas

• Encapsulation for ListNodes

• Implementing size() and re-implementing append()

• invariants are properties that are always true
The *append* method

```java
public class ListNode {
    private int data;
    private ListNode next;

    public void append(int d) {
        if (next == null) {
            next = new ListNode(d);
        } else {
            next.append(d);
        }
    }
}
```

How does the append method traverse (i.e. walk node to node) the linked list?

a) line 18: the Java keyword “next” takes us to the following node in a linked list
b) line 18: by calling append again, it will affect a different ListNode than before
c) line 18: calling append on a different value of d
d) line 18: next looks at the reference to the following ListNode, the dot follows the reference to the actual ListNode object, then we call append on it
e) line 16: assigning next to a new ListNode brings us to the following ListNode
Iterative (for-loop) implementation of append

```java
public void append(int d) {
    ListNode current = this;
    while(current.next!=null) {
        current = current.next;
    }
    current.next = new ListNode(d);
}
```
Method to get length of the list

```java
/*
Return the number of nodes in this list */

public int length() {

100 -> 200 -> 300 \n
length() → 3
```
If it takes 1ms to find the length of a list length 10, how long for a list of size 10,000?

a) 1ms  
b) 1,000ms  
c) 2,000ms  
d) 10,000ms  
e) 20,000ms
Some problems with ListNode

• We have to go through the whole list to **append** a new element

```java
class ListNode {
    int val;
    ListNode next;
}
```

```java
public void append(int d) {
    if (next == null) {
        next = new ListNode(d);
    } else {
        next.append(d);
    }
}
```

• We have to go through the whole list to get the length

```java
/*
 * Return the number of nodes in this list
 */
public int length() {
    if (next==null) { return 1; } 
    else return 1 + next.length();
}
```
A new class, LinkedList

LinkedList uses the ListNode class in its implementation

```java
LinkedList mylist = new LinkedList();
mylist.append(100); mylist.append(200); mylist.append(300);
```
What should be the type for head and tail?

A) int
B) int[]
C) ListNode
D) ListNode[]
E) LinkedList

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
New algorithm for append()?

example usage

```java
LinkedList mylist = new LinkedList();
...
mylist.append(400);
```

1 sentence answer (“clif notes” version)

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
public class LinkedList {
    private ListNode head;
    private ListNode tail;

    public void append(int d) {
        // if (tail.next!=null)
        ListNode n = new ListNode(d);
        tail.next = n;
        tail = n;
    }

    // we also made ListNode an “inner class” of LinkedList to have access to
    // its private fields

    private class ListNode {
        ...
    }
}
Make \texttt{length()} faster, too

\begin{itemize}
  \item mylist
  \item head
  \item tail
  \item \texttt{data next}
  \item \texttt{len}
\end{itemize}

\begin{minipage}{\textwidth}
\begin{itemize}
  \item \texttt{100}
  \item \texttt{200}
  \item \texttt{300}
\end{itemize}
\end{minipage}
(Code)
Today’s big ideas

• Encapsulation for ListNodes

• Implementing size() and re-implementing append()
  • by encapsulating ListNodes within a LinkedList class, we can store references to the head and tail

• invariants are properties that are always true
What to do now

• Quiz 2 then HW 2

• DYB later today

• Collect your Change of Registration form at the front right now if you’ve submitted one