Assignment 3
Linked Lists

Submitting Assignments

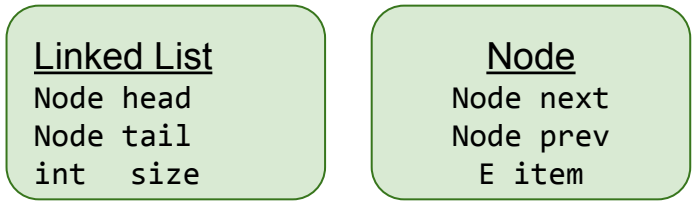
- You ***MUST*** include your source code!!!
  - This means you *.java files
- Follow the instructions in the assignment
  - Use the class names specified in the assignment
  - Methods must have the same “signature”
    - Same parameter types
    - Same return values
- Double check what you’re submitting
  - From the command line:

```
$ jar tf assignment.jar
```

  - Using Eclipse, export your assignment, then import into another project

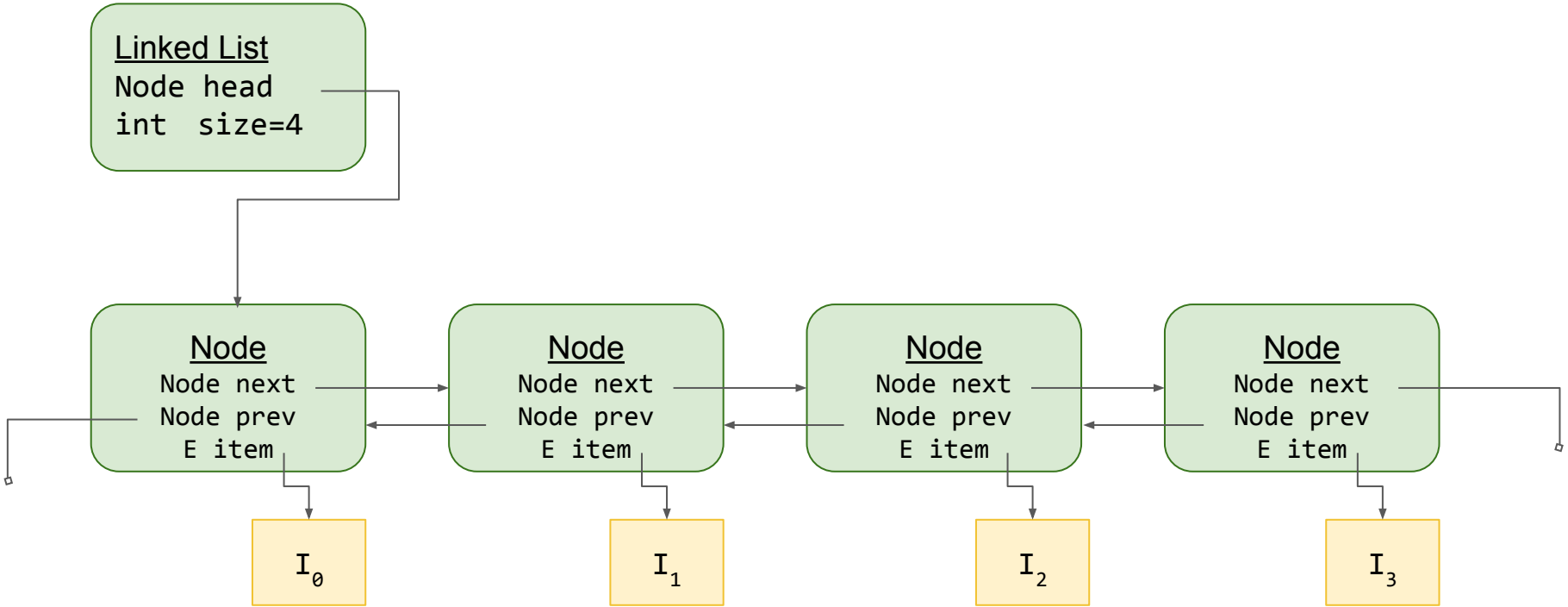
Overview

- The same as assignment 2 except with linked lists
  - `MyLinkedList` still implements the `List211` interface
  - All the methods in this interface should behave identically to those in A02
- Sorting algorithms are conceptually the same, but differ slightly in practice:
  - You can't use indices to directly access element
  - You need to traverse the list using the next/prev member variables of a node
  - Swapping elements also looks a little different
- Your existing `ContactList` class can be used with minimal changes

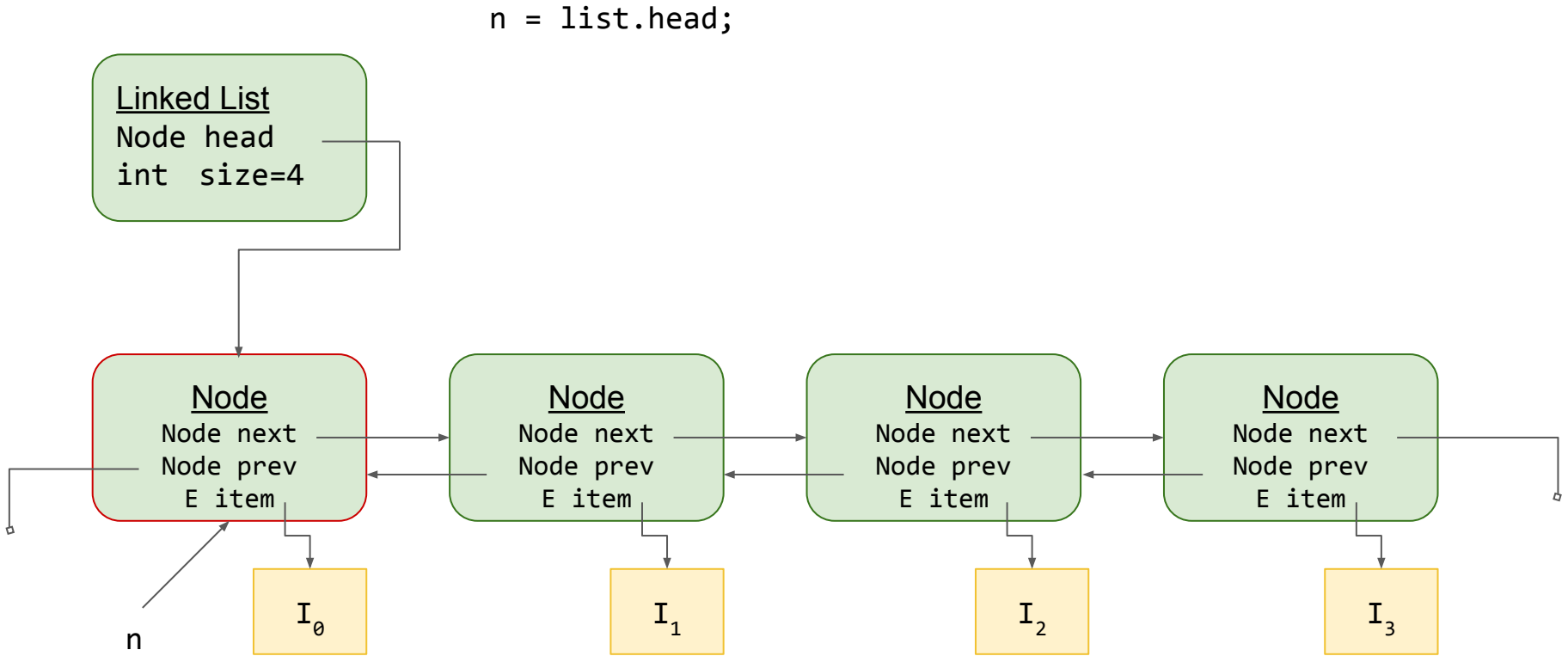


Example

- The “tail” field omitted for simplicity

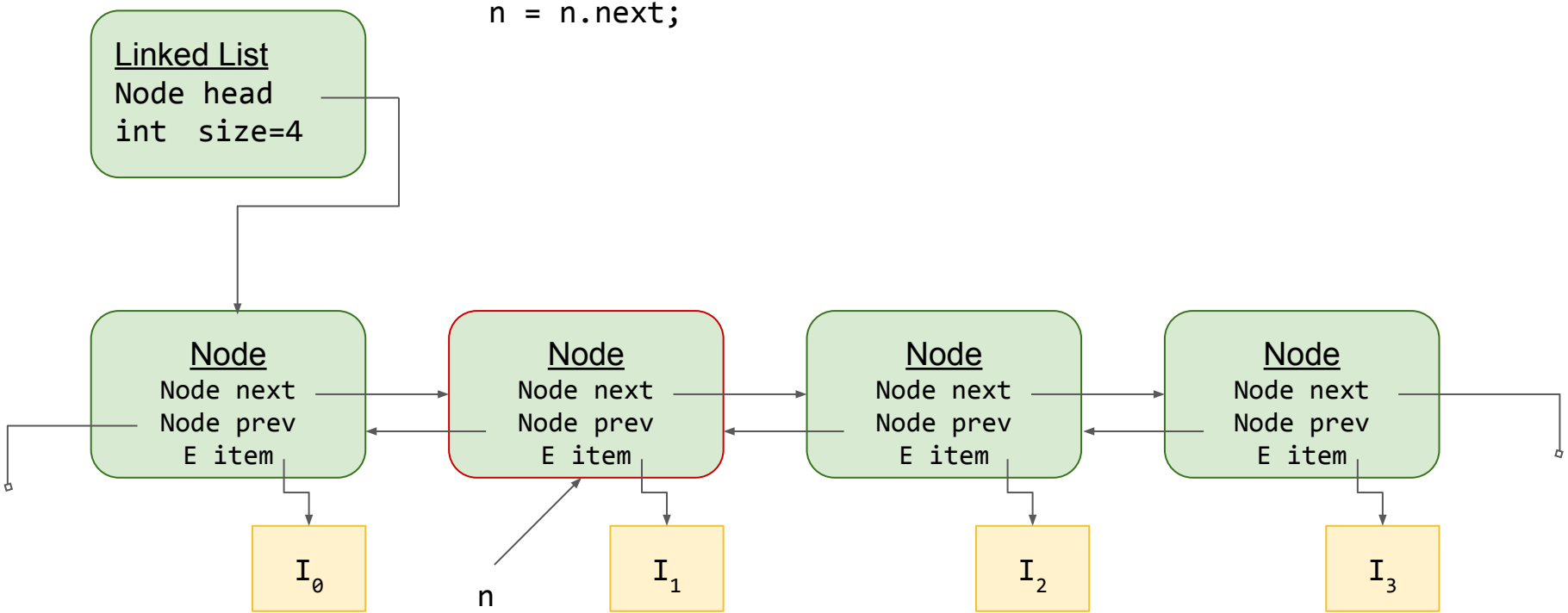


Example


n = list.head;
Linked List
Node head
int size=4
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
n
I0
I1
I2
I3

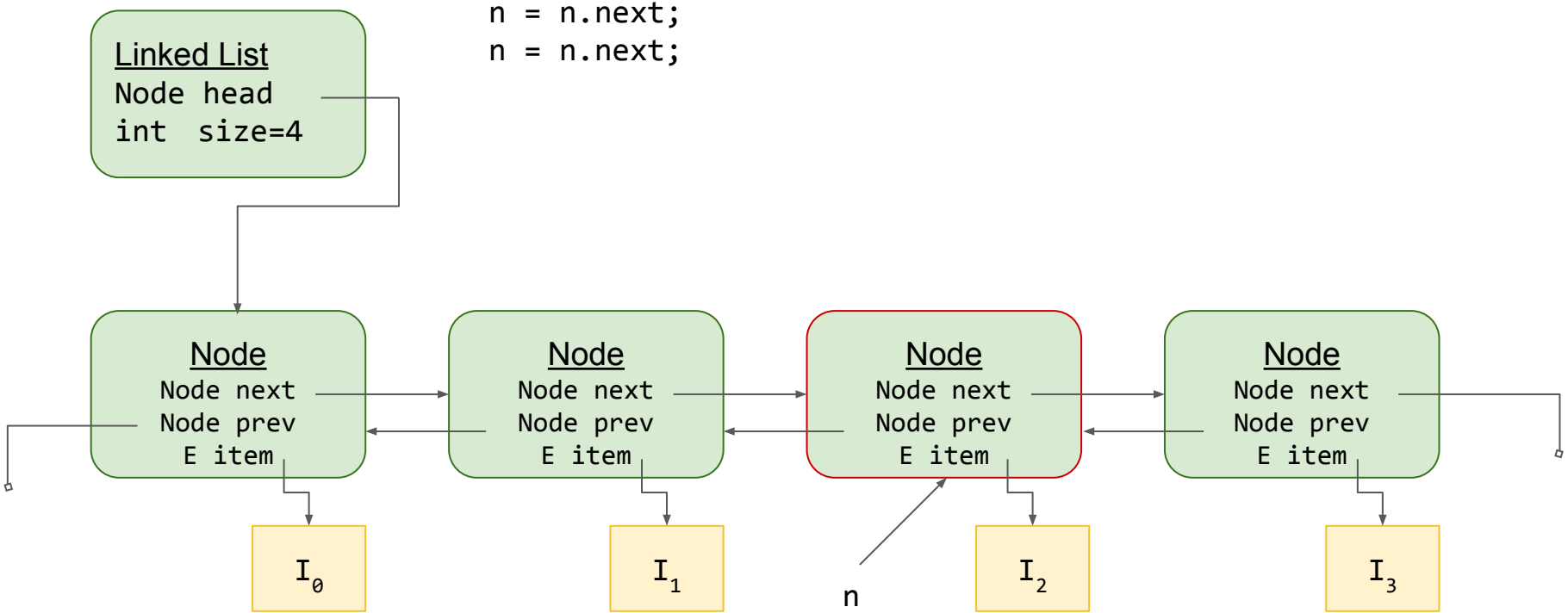
Example

```
n = list.head;
n = n.next;
```



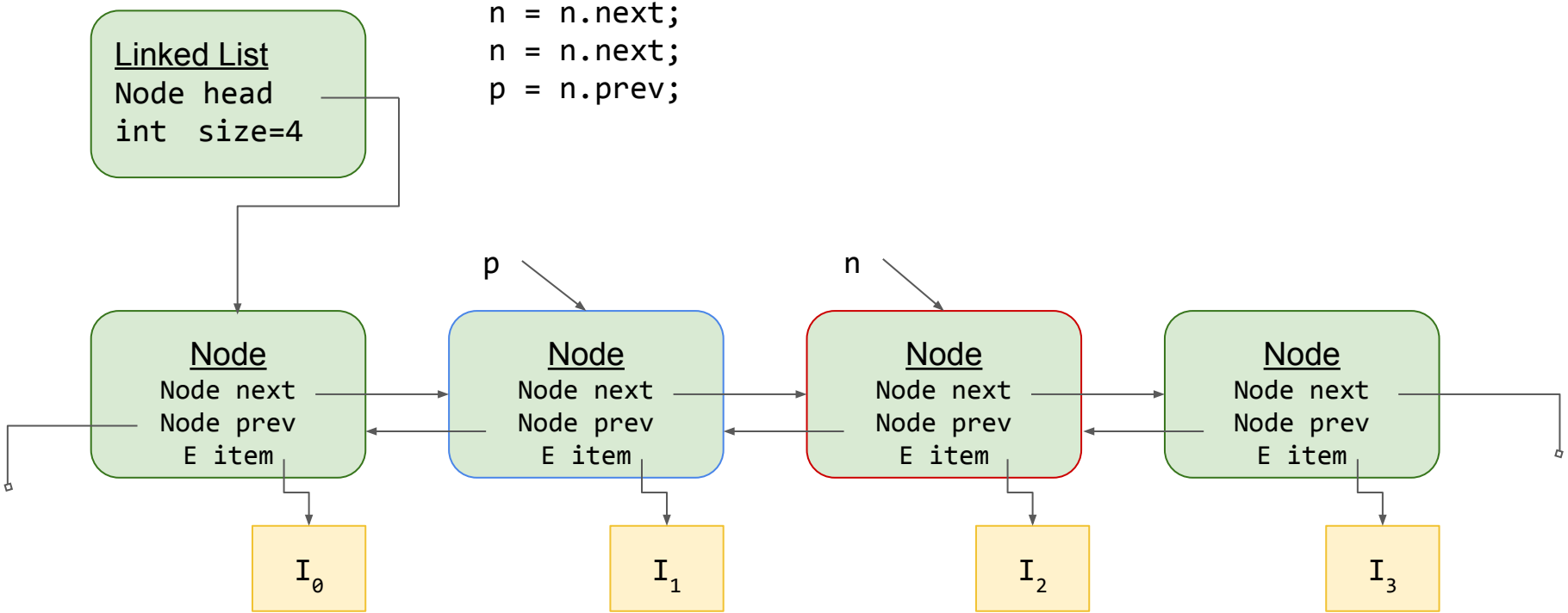
Example

```
n = list.head;
n = n.next;
n = n.next;
```



Example

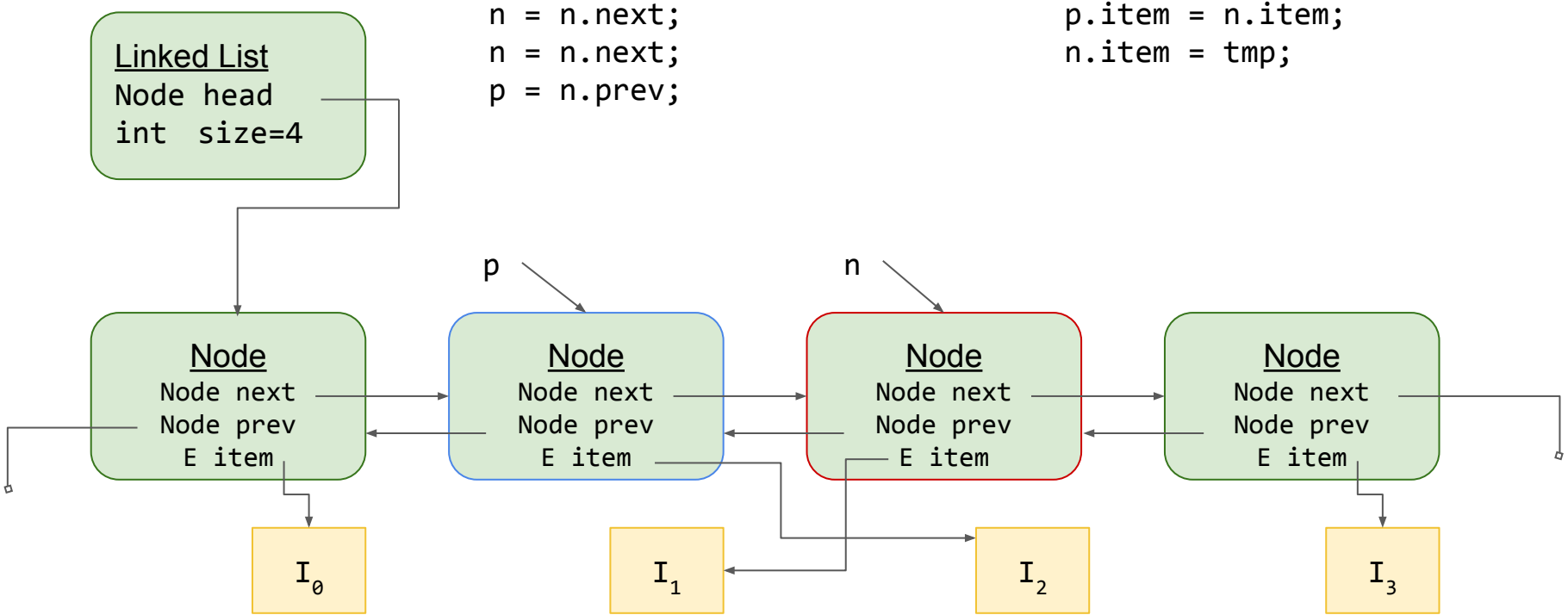
```
n = list.head;
n = n.next;
n = n.next;
p = n.prev;
```



Example

```
n = list.head;
n = n.next;
n = n.next;
p = n.prev;
```

```
tmp = p.item;
p.item = n.item;
n.item = tmp;
```



Extra Credit #1

```
public interface List211<E> {
  boolean add(E e);
  void add(int index, E element);
  E remove(int index);
  E get(int index);
  E set(int index, E element);
  int indexOf(Object obj);
  int size();
}
```

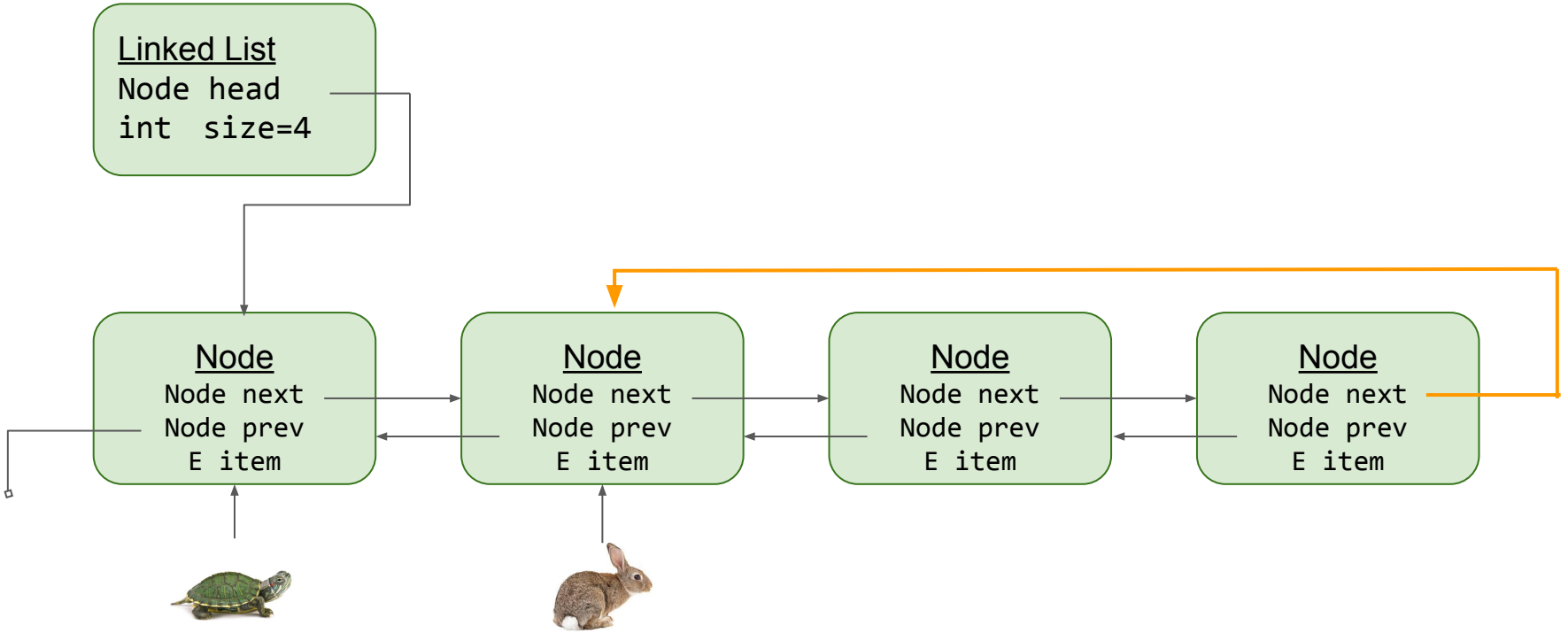
Announcements

- Attendance - 2 absences allowed
- Extra credit
  - Last weeks EC worth 5 points
  - Extra credit will will add up to a possible 5% of overall grade
  - EC is “all or nothing”
  - Don’t wait until the end of the semester
- Follow assignment and submission instructions
  - Code must be implemented as per assignment instructions (class names, interfaces, etc.)
  - Code must compile without errors
  - Submit JAR files (not ZIPs, tarballs, separate source files, etc.)
  - Failure to follow these instructions will result in a ***5 point deduction***
- Starting with assignment 3, submissions with missing files will ***get no credit***!!!

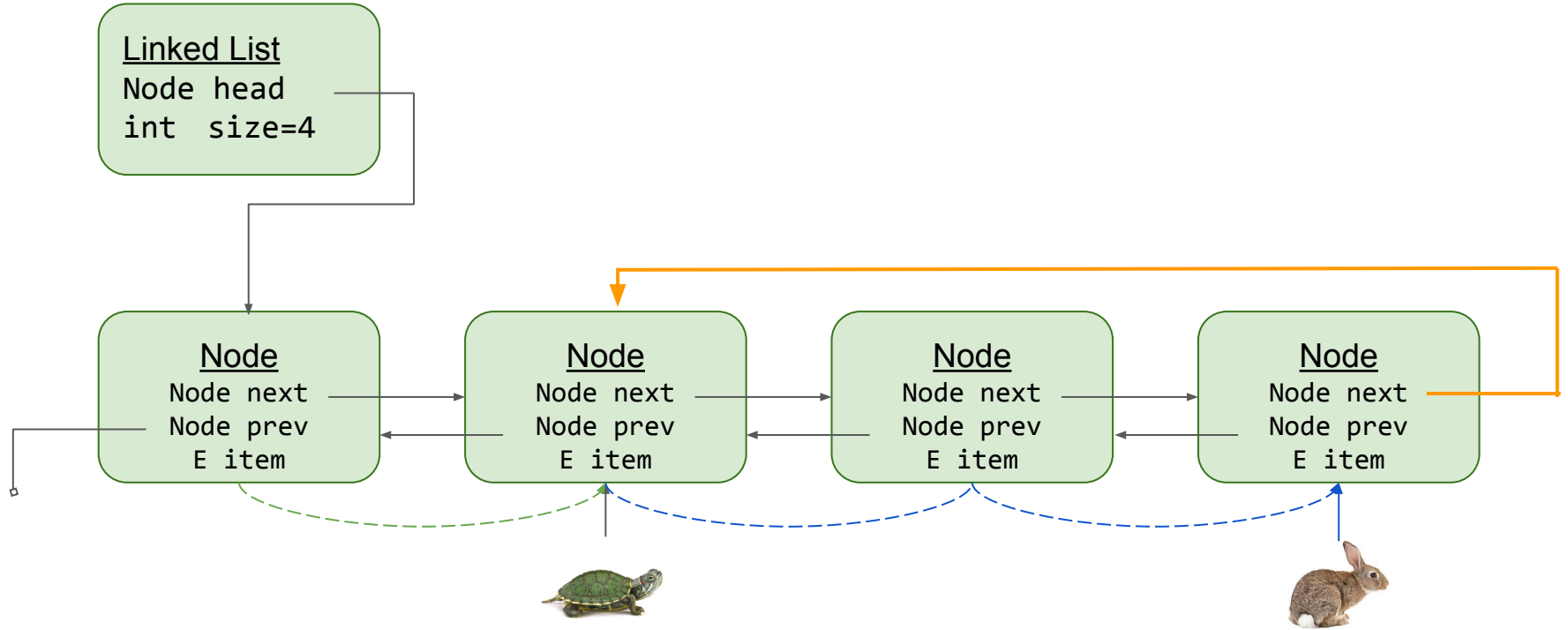
Extra Credit #2 (A03) - Detecting a Cycle

- Worth 15 points (all or nothing)
- A cycle occurs if the current node can be reached again by traversing a link exiting the current node
  1. Doubly linked lists always have cycles (`next.prev`)
  2. For the extra credit, only consider the `next` link
- Algorithm (tortoise and hare):
  1. “Tortoise” pointer starts at first node (head)
  2. “Hare” pointer starts at second node (head.next)
  3. On each iteration, tortoise moves one node, hare moves two
  4. Keep looping until one of the following conditions becomes true:
     a. Tortoise becomes null (no cycle)
     b. Tortoise == Hare (has a cycle)

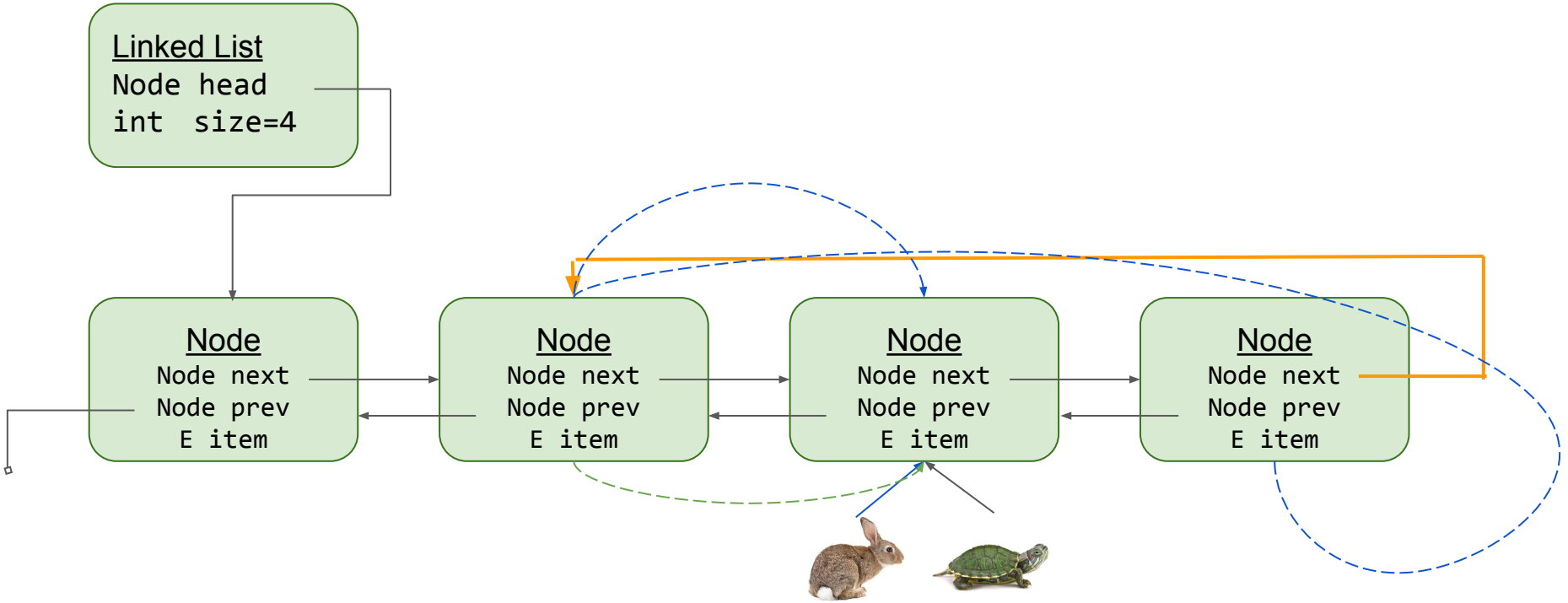
Example


Linked List
Node head
int size=4
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item

Example


Linked List
Node head
int size=4
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item

Example


Linked List
Node head
int size=4
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item
Node
Node next
Node prev
E item

Methods to Implement for MyLinkedList

- Implement the following methods:

```
/**
 * Adds a cycle from the tail node to the node at index “cycleTo”
 * @param cycleTo Changes the “next” field of the node referenced by the tail member
 *        to point to the node at position “cycleTo”.
 */
void addCycle(int cycleTo);

/**
 * Determines if a cycle exists
 * @return Returns true if a cycle exists, false otherwise
 */
boolean hasCycle()
```