First Quiz - Version A
CS 1102 Computer Science 2

Spring 2021

Tuesday March 2, 2021
Instructor Muller

KEY

Before reading further, please write your name on the top of all of your quiz answer sheets.

This is an open notes and open book quiz. But **collaboration is expressly prohibited**.

- Partial credit will be given so be sure to show your work.

| Problem | Points | Out Of |
|---|---|---|
| 1 | | 4 |
| 2 | | 6 |
| Total | | 10 |

- Some variations of this quiz have problems for which you may want to use an ADT such as a `Stack<T>`, `Queue<T>` or `Deque<T>`. Feel free to assume that the `Deque<T>` ADT is available, implemented with a class `ArrayDeque<T>`.
- Some of the problems on the quiz variations involve linked-lists of nodes as shown on the right.
- We'll use the standard Java notation `{ 1, 2, 3 }` for an integer array and we'll use the notation `[ 1, 2, 3 ]` for a linked-list of integers.
- There is no need to be concerned with visibility attributes such as `public` or `private` on this quiz.
- Feel free to write helper functions if you need them.
- **Please write neatly.**

---

```
interface Deque<T> {
  void pushLeft(T item);
  T popLeft();
  void pushRight(T item);
  T popRight();
  int size();
  boolean isEmpty();
  }

class ArrayDeque<T>
  implements Deque<T> {
  ...
}
```

```
static class Node {
  int info;
  Node next;
  // assume any Node constructors that you want
}
```

```
+--------+--------+
|  info  | next o-+-->
+--------+--------+
```

```
// [ 1, 2, 3 ]
new Node(1, new Node(2, new Node(3, null)))
```

```
    +---+---+     +---+---+     +---+---+
o-->| 1 | o-+---> | 2 | o-+---> | 3 | o-+-+
    +---+---+     +---+---+     +---+---+ =
```

## Problem 1: 4 Points

Write a function `static Node add(Node a, Node b)` such that for a call `add(a, b)`, the `add` function returns a `Node` list representing the pairwise sum of `Node` list `a` and `Node` list `b`. For example, with `Node` lists:

```
Node ms = new Node(10, new Node(20, new Node(30, null)));  // [ 10, 20, 30 ]
Node ns = new Node( 5, new Node( 5, null));                // [  5,  5  ]
```

the call `add(ms, ns)` should return the linked list of nodes `[ 15, 25, 30 ]`. Note that the inputs need not be of the same length and either or both inputs could be `null`.

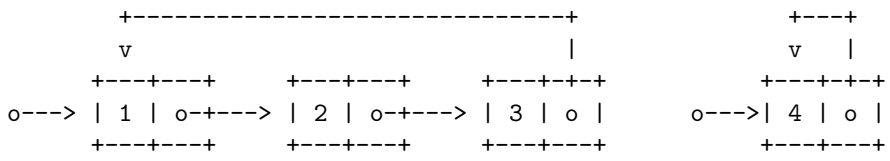
**Answer:**

```
// 1.1: add two linked lists of ints, return a linked list
//
private static Node add(Node a, Node b) {
  if (a == null && b == null) return null;
  if (a == null) return b;
  if (b == null) return a;
  return new Node(a.info + b.info, add(a.next, b.next));
}
```

## Problem 2: 6 Points

Solve **only one** of the following two problems.

1. A linked list of of `Nodes` has a *cycle* if the `next` field of any `Node` in the list points to a previous `Node` in the list. Examples include:



   Write a function `static boolean isCyclic(Node a)` such that a call `isCyclic(a)` returns `true` if `a` has a cycle. The function should return `false` if the list has no cycle or if it is `null`.

   **Answer:**

```
// 2.1A: return true if a linked list of nodes has a cycle

// Recursive solution
//
private static boolean cycleHere(Node a, Node b) {
  if (b == null)
    return false;
  else
    return (a == b) || cycleHere(a, b.next);
}

public static boolean isCyclic(Node a) {
  if (a == null)
    return false;
  else
    return cycleHere(a, a.next) || isCyclic(a.next);
}

// Iterative solution
//
public static boolean isCyclic(Node a) {
  if (a == null) return false;
  while (a != null) {
    Node b = a.next;
    while (b != null) {
      if (b == a) return true;
      b = b.next;
    }
    a = a.next;
  }
  return false;
}
```

2. Write a function `static int[] insert(int k, int[] a)`. You may assume that the array `a` is non-empty and in strictly ascending order. The `insert` function should return a new array, just like `a`, but the integer `k` has been inserted in the appropriate position. The result array should be of length `a.length + 1`.

**Answer:**

```
// 2.1B: insert an int into an ascending sorted array
public static int[] insert(int k, int[] a) {
  int[] b = new int[a.length+1];
  int i = 0;
  while (a[i] < k) b[i] = a[i++];
  b[i] = k;
  while (i < a.length) b[i+1] = a[i++];
  return b;
}
```