First Quiz - Version B
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 append(Node a, Node b)` such that for a call `append(a, b)`, the `append` function returns a `Node` list representing the result of appending `a` and `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 `append(ms, ns)` should return the linked-list of nodes `[ 10, 20, 30, 5, 5 ]`. Note that either or both inputs could be `null`. And *after* such a call, the node lists `ms` and `ns` remain as they were before the call.

Answer:

```
// 1.2: append two linked lists
public static Node append(Node a, Node b) {
  if (a == null)
    return b;
  else
    return new Node(a.info, append(a.next, b));
}
```

Problem 2: 6 Points

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

1. A linked list of of `Node`s is *palindromic* if it is `null`, of length 1 or if the first and last elements are equal and the linked list omitting the first and last elements is palindromic. For example, all of `[ 3 ]`, `[ 4, 4 ]` and `[ 5, 6, 5 ]` are palindromic but `[ 4, 5 ]` is not.

   Write a function `static boolean palindromic(Node a)` such that a call `palindromic(a)` returns `true` if `a` is palindromic. The function should return `false` if `a` is not palindromic.

   Answer:

```
// 2.2A: palindromic int linked list
public static boolean palindromic(Node a) {
  Deque<Integer> dq = new ArrayDeque<Integer>();
  while (a != null) {
    dq.addFirst(a.info);
    a = a.next;
  }
  while (dq.size() > 1)
    if (!dq.removeFirst().equals(dq.removeLast())) return false;
  return true;
}
```

2. Write a function `static int[] merge(int[] a, int[] b)`. You may assume that the arrays `a` and `b` are both in strictly ascending order. The `merge` function should return a new array of length `a.length + b.length` with the integers from `a` and `b` in ascending order. For example, with

```
int[] ms = {2, 4, 6};
int[] ns = {1, 3, 5};
```

   the call `merge(ms, ns)` should return the array `{1, 2, 3, 4, 5, 6}`.

   Answer:

```
// 2.2B: merge two ascending sorted int arrays
public static int[] merge(int[]a, int[]b) {
  int[]c = new int[a.length + b.length];
  int i = 0, j = 0, k = 0;
  while (i < a.length && j < b.length)
    c[k++] = a[i] <= b[j] ? a[i++] : b[j++];
  while (i < a.length) c[k++] = a[i++];
  while (j < b.length) c[k++] = b[j++];
  return c;
}
```