Introduction to Comp. Sci.,
Practice Questions for Final Exam

actual final exam: 8am in WWH 109, May 15, 2013

Question 1

Write a static method, `straight`, that takes an array of integers as an argument and returns `true` if the array contains a contiguous range of integers in any order. For example:

```
straight(new int[]{ 5, 7, 6 })   // returns true
straight(new int[]{ 1, 3, 4 })   // returns false
straight(new int[]{ -1 })        // returns true
straight(new int[0])             // returns true
```

You may use any standard Java library class. Don't worry about minor details like `set` vs. `put` if your meaning is clear. The argument may be of any length, including zero, but is guaranteed not to be null.

Question 2

Suppose you have a `Node` object, defined like this.

```
public class Node {
  int value;
  Node a;
  Node b;
}
```

A `Node` has two children, `a` and `b`. Either or both may be `null`. You can assume that no `Node` will have itself as a child, directly or indirectly. A `Node` is a *heap* if and only if both of the following are true:

1. It does not have a child with a smaller `value` field. This is trivially true if both `a` and `b` are `null`.
2. All of its non-`null` children are heaps.

Write a method `isHeap` that returns a `true` if its argument is a heap and `false` if it is not:

```
public static boolean isHeap(Node node) {
  ...
}
```

You can treat `null` arguments however you like.

Question 3

Implement a class that represents an amount of US money in dollars and cents. It should support these operations:

```
public class Money {
  public Money(int dollars, int cents)
      throws IllegalArgumentException;
  public int getDollars();
  public int getCents();
  @Override public String toString();
  @Override public boolean equals(Object that);
  public static Money add(Money x, Money y);
}
```

Both `dollars` and `cents` should be non-negative. The constructor should throw an `IllegalArgumentException` otherwise. `add` should return a newly-allocated `Money` representing the sum of the two arguments. For example, the following code:

```
Money z = Money.add(new Money(5, 10), new Money(3, 92));
System.out.println(z.toString());
```

should output "$9.02". To prevent rounding errors, you should not use floating point numbers in your solution.

Question 4

Write a static method, `reverseComparator`, that, given a `Comparator`, returns another `Comparator` that represents the opposite order. So, if `cmp.compare(a, b)` returns a negative number, then `reverseComparator(cmp).compare(a, b)` should return a positive number. (The logic here is pretty simple, but how do you write the type for this?)