COLBY COLLEGE
Department of Mathematics
and Computer Science

CS231 **Allen B. Downey**
Fall 1998

Assignment 4: Exam Recovery

Due: Monday 12 October
The purpose of this assignment is to pick up the pieces after the first exam.

Recursion

This is a class definition for a Tree data structure, which is similar to a ListNode except that each element of the tree points to *two* other Trees.

```
public class Tree {
  int value;
  Tree left, right;

  public Tree (int value_, Tree left_, Tree right_) {
    value = value_;
    left = left_;
    right = right_;
  }

  public static int totalTree (Tree tree) {
  }

  public static void main (String[] args) {
    Tree root = new Tree (3, null, null);
    Tree branch1 = new Tree (5, null, null);
    Tree branch2 = new Tree (7, null, null);

    root.left = branch1;
    root.right = branch2;

    int x = totalTree (root);
    System.out.println (x);
  }
}
```

1. Type in this class definition (or cut and paste it from the HTML version of this assignment), and then fill in the body of the `totalTree` method so that it adds up the value field from all the elements of a given tree. Hint: this is hard to do without using recursion, and pretty straightforward if you do.

2. Test your code on a variety of trees, including lopsided trees and null trees.
3. Turn in a copy of this method only.

Lists

1. Load up your List class from a couple of assignments ago or get mine from my web page.
2. Add a method to the List class called `noLoops` that checks whether the List contains a loop. Note that the loop does not necessarily go back to the first element, so you have to make sure that you never see *any* nodes more than once.
3. Convince yourself that your method will terminate even if the List contains a loop, and test it with a couple of Lists that do and do not contain loops.
4. Turn in a copy of this method only (along with any helper methods you created).

Priority Queue

Assume that you have an implementation of a Priority Queue that provides the usual set of methods with the usual behavior:

```
public class PriorityQueue {
  public PriorityQueue ();
  public int size ();
  public void insert (PQItem newItem);
  public PQItem remove ();
}
```

While working on another program, it occurs to you that it would be useful to have a new method called `removeNthElement` that would take a number n as an argument and would remove and return the element with the nth highest priority.

You have a choice between two approaches:

internal: you can modify the file PriorityQueue.java and add the new method to the class definition. In this approach, you can access the instance variables as well as the other methods.

external: you can build the new method externally (in another file), using only the publicly-accessible methods and not the instance variables.

1. Using *any of the implementations* we have seen so far, write an external implementation of `removeNthElement`. Even thought it is an external method in the sense defined above, you can write it inside PriortityQueue.java. For the sake of clarity, though, you should put it down at the bottom with `main`. Also, make it a class method, so it takes the queue and n as arguments.
2. Add code to `main` to test your program.
3. Copy your code from this implementation into another and confirm that it still works, without modification.
4. Turn in a copy of this method only, along with any comments about your experience.