15. Given the node class

```
public class Node {
     private int datum;
     private Node next;
     public Node(int d, Node n)  {
         datum = d;
         node = n;
    }
    public Node next() { return next; }
    public int datum() { return datum; }
    public void setNext(Node next) { this.next = next; }
}
```

complete the following list class. If the list is empty, `removeHead()` is undefined (that is, you are not responsible for that case; it is ok, for example, if an exception is thrown.) (16 points.)

```
public class List {
     private Node head;
     public List() { head = null; }

     public void addToFront(int item) {      // add a new element at front



     }

     public void removeHead() {       // remove the first element



     }

     public int average() {       // find the average of all elements









     }
}
```

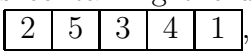
16. Given the following `Node` class for a recursive version of a list, write the method `contains` which will return `true` if this node or any node following it contains the given `item` (that is, `true` if `item` is in the list), and `false` otherwise. *You may not test if any reference is null.* (Hint: Use exception handling.) (10 points.)

```
public class Node {

        private int datum;
        private Node next;

        public boolean contains(int item) {









        }
}
```

17. Write a class that will emulate a stop watch. For example, when you turn the stop watch on, it initially reads 0 seconds. Then you press the start button, and after 3 seconds, you press the stop button. It now reads 3. Pressing the stop button again, when it is already stopped, does nothing. Your press the start button a second time and let it run for 5 seconds (pressing it a third time, while it is already running, does nothing). When you press the stop button again, it reads 8 seconds (3 + 5). Your class should have methods `void start()`, `void stop()`, and `int getTime()`. The method `getTime()` should return the total elapsed milliseconds that the watch had been running. The class should work in the driver to the left.

```
import java.util.Scanner;
public class SWDriver {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);
      StopWatch sw = new StopWatch();
      System.out.println("What is your name?");
      sw.start();
      String name = keyboard.nextLine();
      sw.stop();
      System.out.println("How old are you?");
      sw.start();
      String age = keyboard.nextLine();
      sw.stop();
      System.out.println("What is your quest?");
      sw.start();
      String quest = keyboard.nextLine();
      sw.stop();
      System.out.println(name + ", it took you an ave of "
                         + (sw.getTime() / 3.0)
                         + " ms to answer each question.");
  }
}
```

Hints: Use the method `System.currentTimeMillis()` returns the current state of the computer's clock (number of milliseconds since midnight, Jan 1, 1970); assume `System.currentTimeMillis()` returns an `int`. Your class will have to model the fact that a stop watch can be in either a "running" or "not running" state. (12 points)

18. a. Write a method which, given an array of `int`s containing the digits of a number will return that integer. For example, given 

, it would return 25341. (8 points)

b. Write a method which, given a `String` containing an integer, will return the equivalent `int`. For example, given `"25341"`, it will return 25341. You may assume the `String` you are given is correct—it contains an integer and only an integer. Hint: Recall arithmetic operations on `char`s that you did in the Caesar cipher projects. The codes for the digits are consecutive in the order you would expect them, 0123 etc. (8 points)

20. Write a class that keep track of the check-out history of books and patrons in a library. Every time a book is checked out, this is reported to an object of this class using the method `checkedOut()`. The method `booksCheckedOut()`, given the name of a patron, will return a `String` listing the books the patron has checked out. The method `patronsCheckedOutBy()`, given the name of a book, will return a `String` listing the patons who have checked out this book. Your class should implement the following interface

```
interface LibraryRecord {
     void checkedOut(String patron, String book);
     String booksCheckedOut(String patron);
     String patronsCheckedOutBy(String book);
}
```

Hint: Use two `HashMap<String,String>`s, whose interface is found on the next page. (12 points)

```
public class HashMap<String> {

        // Test if this map has a value associated with a given key
        public boolean containsKey(String key);

        // Retrieve the value associated with a given key
        // (null if none)
        public String get(String key);

        // Associate a given value with a given key
        public void put(String key, String value);
}
```