Programming Assignment 8-2

Use the class `MyStringLinkedList` in the source folder for this lab as a starting point for implementing a doubly linked list (with header) for use with `String` data.

Implement the operations:

```
//inserts a new Node containing data so that its
//position in the list is now pos
void insert(String data, int pos)

//attempts to remove the first Node that contains
//data; if successful, returns true; otherwise, false.
boolean remove(String data)

//recursively attempts to find a String in some Node in the
//list; returns true if found, false otherwise
boolean recurSearch(String data)
```

Also, implement the methods in `MinSort` and `Search` in this new context. Test your sort and search methods as in 8-1, using the following:

a. Sort the following list

```
["big", "small", "tall", "short", "round", "square",
"enormous", "tiny","gargantuan", "lilliputian",
"numberless", "none", "vast", "miniscule"]
```

b. Take the list sorted in part a. and attempt searches for each of the following:
   - `"number"`
   - `"tiny"`

Hints:

1. For part a, replace the `swap` method for arrays with

   ```
   void swap(Node n1, Node n2),
   ```

   which *appears* to switch the positions of `n1` and `n2` in the list with the following trick: it switches the *values stored* in the two nodes. (It is possible to actually swap the positions of the nodes by rearranging links, but it is tricky and not needed for this lab.)

   And replace the `minpos` method (which finds the position of the min value) with

   ```
   Node minNode(Node n)
   ```

   which returns the `Node` nested in `n` that has the minimum value.

2. For part b, you can re-do the binary search algorithm in the context of list Nodes – there is no clever optimization as there is for sorting. Notice how costly it is to locate the Node in the middle.