COMS 3134 Homework 2

Submission instructions

All programs must compile and run on CUNIX to receive credit. Submit your electronic files via `http://courseworks.columbia.edu`. We prefer electronic submission of theory, though you will not be penalized for paper submissions. (Do **not** print out your programs.) I recommend learning LaTeX for typesetting math. Include a README file that explains exactly what each file in your submission is. Place all the files you want to submit into a submission directory with the following naming scheme.

`<your_uni>_hw<number>`

So if my UNI is `uni1234` am submitting homework 6, my directory would be `uni1234_hw6`. Archive your submission directory using

```
tar -czvf uni1234_hw6.tar.gz uni1234_hw6
```

and upload `uni1234_hw6.tar.gz` to courseworks. (You will probably need to first download the file to a local directory using an FTP program. See CUNIX tutorial for more info.)

Multiple Submissions: You can submit multiple times, but we will only consider the latest submission based on the timestamp in courseworks. Please give at least 1-2 minutes between two submissions so we can tell which is the newest submission.

I recommend that you also keep a pristine copy of your submission folder in case there is any submission error.

Theory Problems

1. (Weiss 3.2) **4 points.** [Describe the algorithm to s]wap two adjacent elements by adjusting only the links (and not the data) using:
   (a) Singly linked lists.
   (b) Doubly linked lists.
2. (Weiss 3.8) The following routine removes half of the list passed as a parameter:

```
public static void removeFirstHalf( List<?> lst ) {
    int theSize = lst.size() / 2;
    for ( int i = 0; i < theSize; i++ )
        lst.remove( 0 );
}
```

(a) **2 points.** Why is `theSize` saved prior to entering the `for` loop?

(b) **6 points.** What is the running time of `removeFirstHalf` if `lst` is an `ArrayList`?

(c) **6 points.** What is the running time of `removeFirstHalf` if `lst` is a `LinkedList`?

(d) **2 points.** Does using an iterator make `removeFirstHalf` faster for either type of `List`?

3. (Weiss 3.34) A linked list contains a cycle if, starting from some node p, following a sufficient number of `next` links brings us back to node p. Node p does not have to be the first node in the list. Assume that you are given a linked list that contains N nodes. However, the value of N is unknown.

   (a) **6 points** Design an $O(N)$ algorithm to determine if the list contains a cycle. You may use $O(N)$ extra space.

   (b) **Extra credit 3 points.**[1] Repeat part (a), but use only $O(1)$ extra space. (*Hint*: Use two iterators that are intially at the start of the list, but advance at different speeds.)

4. **4 points.** (Weiss 3.36) Suppose we have a reference to a node in a singly linked list that is guaranteed *not to be the last node* in the list. We do not have references to any other nodes (except by following links). Describe an $O(1)$ algorithm that logically removes the value stored in such a node from the linked list, maintaining the integrity of the linked list. (*Hint:* Involve the next node.)

[1] Extra credit policy: extra credit points only apply as *makeup* points, and cannot give you more than the total number of standard points. This way, students who score 100% on the standard credit problems but do not do the extra credit are not penalized.

Programming

For this programming assignment, you will be working with a few more .java files than in Homework 1, so please use a makefile to keep things organized. Here is a sample makefile that you can copy and modify as needed (if you use other .java classes). For any classes described in the assignment, use the filenames and naming conventions we provide for uniformity.

`http://www1.cs.columbia.edu/ bert/courses/3134/hw2/Makefile`

1. (Weiss 3.28) **15 points.** A **deque** is a data structure consisting of a list of items, on which the following operations are possible:

   - `push(x)`: Insert item `x` on the front end of the deque.
   - `pop()`: Remove the front item from the deque and return it.
   - `inject(x)`: Insert item `x` on the rear end of the deque.
   - `eject()`: Remove the rear item from the deque and return it.

   Each of these operations should execute in $O(1)$ time.

   Name your deque class `MyDeque` and use the exact method names described above. Your class should be generic and may use either the built in Java `ArrayList/LinkedList` classes or the textbook's `MyArrayList/MyLinkedList` implementations. (*Hint*: you should only need one of these.) Alternatively, you can also build this data structure from scratch, without piggybacking on these other classes.[2] Test your class using the class `MyDequeTester`, which you will find on the course homepage.

2. **Palindrome Detector (15 points)**. A palindrome is a phrase that reads the same forwards as it does backwards. For example, "a man, a plan, a canal, Panama," is a palindrome. Write a program that uses a stack to check for palindromes in each line of a text file. Try your program on the example text file,

   `http://www.cs.columbia.edu/~bert/courses/3134/hw2/palindromes.txt`

   Your program should output the palindromes that it finds in the document. For example:

   ```
   java PalindromeFinder palindromes.txt
   "a man, a plan, a canal, Panama" is a palindrome.
   "Don't nod" is a palindrome.
   "Taco Cat!" is a palindrome.
   ...
   ```

   You may write a MyStack class for this problem, or you may use your MyDeque class since it supports the two stack methods. Don't use the built in Stack. Feel free to use either a LinkedList, ArrayList or an array to implement MyStack.

[2] This may be the optimal choice in many cases, because a general list class contains much more code than is necessary to implement the operations of a deque. However, in this case, you don't have to write any of the list code, so using a list class isn't so bad.