CS2134 HOMEWORK 7
Due* Sunday Nov 6, 2016 at 11:00 p.m.

Be sure to include your name at the beginning of each file! Assignment 7 includes a programming portion and a written part. The programming portion must compile and consist of a single file (hw07.cpp). The typed portion should consist of a single file (hw07written.pdf). It *must* be in a .pdf format. Be sure to include your name at the beginning of each file! You must hand in the file via NYU Classes.

Programming Part:

1. Add the following methods[1] to the List class:
   (a) The copy constructor, `List( const List & rhs )`. This method must take `O(n)` time.
   (b) The destructor, ~`List( )`. This method must take `O(n)` time.
   (c) The method `front( )`. It performs as stated in `www.cplusplus.com/reference/forward_list/forward_list/front/` This method must take O(1) time.
   (d) The method `merge( List & alist)`. It performs as stated in `www.cplusplus.com/reference/forward_list/forward_list/merge/`
   (e) The method `remove_adjacent_duplicates( )`. The method removes any element if it is adjacent to a node containing the same item[2]. Thus if the list contained $a, a, a, b, b, c, c, c, c$ afterwards it would contain a, b, c. If the list contains a, b, a, b, a then afterwards it contains a, b, a, b, a (i.e. no change, since no items adjacent to each other were the same).
   (f) The method `remove_if( Predicate pred )` that performs as stated in `www.cplusplus.com/reference/forward_list/forward_list/remove_if/` This method must run in `O(n)` time. Your method *should* call your method `erase_after`.

   A simplified version of the `List` class is in a file `simple-list.cpp`. We will test your code by including it in a driver program that uses the methods you implemented. You must test your code by writing and executing your own driver. Remember to hand in your driver code for any programming assignment.
2. Efficiently implement[3] a queue class called `Queue` using a singly linked list, with no header *node* or tail *node*(i.e. nodes that contain no data). Your class should have the methods: `front, back, empty, enqueue, dequeue`.
3. (Extra Credit) Suppose that a doubly linked list class, `Dlist` is implemented with both a head and a tail node. Write a method[4] to remove all nodes containing `x`.

```
template<class Object>
void DList::remove(const Object & x)
```

A simplified version of the `DList` class is in a file `simple-doubly-linked-list.cpp`.

*5% extra credit will be given if you turn this assignment in on Saturday Nov 5 at 11:00 p.m.
[1]Do written question 2 before this question in order to get experience writing pseudo code.
[2]If the list was sorted, the list would remove all duplicates.
[3]Please do written question 7 first.
[4]The `erase` method is *not* included in the simplified doubly linked list class I uploaded. I did not include it so you would get to practice working with pointers for a doubly linked list class. You may write your *own* `erase` method.

Written Part:

1. For the programming questions in 1 (except 1a)
   - draw pictures showing how the links change (or don't change) for programming questions
   - provide the pseudo code for the methods
2. For the `static array` implementation of the `ADT stack` with array size `MAX = 4`, show the conceptual representation of the contents of the stack for each line of code below:

```
Stack<char> s;
s.push('a');
s.push('b');
s.push('d');
s.pop();
s.push('c');
s.pop();
s.pop();
```

3. For the `linked list` implementation of the `ADT stack`, show the conceptual representation of the contents of the stack for each line of code below:

```
Stack<char> s;
s.push('a');
s.push('b');
s.push('d');
s.pop();
s.push('c');
s.pop();
s.pop();
```

4. For your `linked list` implementation of the `ADT queue`, show the conceptual representation of the contents of the stack for each line of code below:

```
Queue<char> q;
q.enqueue('a');
q.enqueue('b');
q.enqueue('d');
q.dequeue();
q.enqueue('c');
q.dequeue();
q.dequeue();
```

5. For your `dynamic array` implementation of the `ADT queue` where it initial size of the array is `4`, show the conceptual representation of the contents of the stack for each line of code below:

```
Queue<char> q;
q.enqueue('a');
q.enqueue('b');
q.enqueue('c');
q.dequeue();
q.enqueue('d');
```

```
q.dequeue();
q.dequeue();
q.enqueue('e');
q.enqueue('f');
q.enqueue('g');
q.enqueue('h');
```