Data Structures – Lab 4
S. Miller

Lab #4

This program tests the concepts of:

- Recognize the advantage of Doubly Linked Lists.
- Learn the basic operations on Doubly Linked Lists.

Objective:

Doubly Linked lists are used to store collections of data. Doubly linked list remove some of the shortcomings in basic linked lists. However, their setup requires additional management of pointers. In addition, a doubly linked list usually has a pointer pointing to the head of the list, and another pointing at the last node in the list.

Add the following two functions to linkedlistd.cpp.

```
//void appendAt( const ListNode::value_type& newdatum,ListNode* cursor )
//     Precondition: cursor is not NULL
//     Postcondition: A new node is created with the datum of newdatum.
//           The new node next points to cursor->next
//           The new node prev points to cursor
//           The cursor->next prev points to the new node
//           The cursor next points to the new node
//           Tail is updated when a node is added to the end
//           Returns a pointer to the new node
ListNode* LinkedList::appendAt (const ListNode::value_type&,ListNode* = NULL)


//void insertAt( const ListNode::value_type& newdatum,ListNode* cursor = NULL )
//     Precondition: none
//     Postcondition: A new node is created with the datum of newdatum.
//           The new node next points to cursor
//           The new node prev points to cursor->prev
//           The cursor->prev next points to the new node
//           The cursor prev points to the new node
//           Head is updated when a node is added to the front
//           Tail is updated when a node is added to an empty list
//           Returns a pointer to the new node
ListNode* LinkedList::insertAt (const ListNode::value_type&,ListNode* = NULL)
```

The same two **`insertItem and makeList`** exist from Lab3, however, the implementation is provided. However, the implementation relies on appendAt and insertAt to function.

The operator << in lablistlistd.cpp is the same as before. However, the operator >> in lablistd.cpp is designed to output the list in reverse order. Both of these are provided so that your output will easily match the solution.

Submission

You should only have to implement linkedlist.cpp for this assignment:

1. lablinklist.cpp is provided and can be used to test your implementation
2. Make sure you have zero memory leaks at program exit

Submitted files need to be named as follows:

File	**Format (lowercase)**	**Example**
Lab4 Implementation	[firstInitial][lastname]_linkedlist.cpp	smiller_linkedlist.cpp

2. **Record Layouts:** None

3. **Special Calculations:** None

4. **Additional Policy:** Make sure you have no memory leaks. Absolutely no #includes in your implementation other then <ostream> and "linkedlistd.h." No recursion!

5. **Output Layout:** Output should look as below:

```
->[5]->[10]->[20]--X

X--[20]<-[10]<-[5]<-

--X

->[30]->[5]->[10]->[20]->[7]->[2]--X

X--[2]<-[7]<-[20]<-[10]<-[5]<-[30]<-

->[1]->[2]->[4]->[8]->[16]->[32]->[16]->[8]->[4]->[2]->[1]--X

X--[1]<-[2]<-[4]<-[8]<-[16]<-[32]<-[16]<-[8]<-[4]<-[2]<-[1]<-
```