Lab 11 Study Questions

1. What is the advantage to using a doubly linked list instead of a singly linked list?

2. What did we add to the LNode struct to implement a double linked list?

3. What did we add to the LList2 class to implement a double linked list?

4. What changes had to be made to the InsertFirst function to implement a doubly linked list?

5. What changes had to be made to the InsertLast function to implement a doubly linked list?

6. What changes had to be made to the DeleteFirst function to implement a doubly linked list?

7. What changes had to be made to the DeleteLast function to implement a doubly linked list?

8. Under what conditions will the following implementation of the equality operator (==) for a doubly linked list cause a Segmentation Fault?
   ```cpp
   template <class LT>
   bool LList2<LT>::operator == (const LList2 & other)
   {
       LNode * n1 = first, n2 = other.first;
       while (n1 != NULL)
       {
           if (n1->data != n2->data)
               return false;
           n1 = n1->next;
           n2 = n2->next;
       }
       return true;
   }
   ```

9. Why will the following implementation of the assignment operator (=) for a doubly linked list likely cause a Segmentation Fault?
   ```cpp
   template <class LT>
   LList2<LT> & LList2<LT>::operator = (const LList2 & other)
   {
       first = other.first;
       last = other.last;
       size = other.size;
       return true;
   }
   ```

10. Why will the following implementations of the InsertFirst and InsertLast functions for a doubly linked list likely cause a Segmentation Fault?
    ```cpp
    template <class LT>
    bool LList2<LT>::InsertFirst (const LT & value)
    {  
    ```
if (size == 0)
    return InsertLast (value);
LNode * n = new LNode;
if (n == NULL)
    return false;
    n->data = value;
n->next = first;
if (first)
    first->prev = n;
else
    last = n;
first = n;
size++;
return true;
}

template <class LT>
bool LList2<LT>::InsertLast (const LT & value)
{
    if (size == 0)
        return InsertFirst (value);
LNode * n = new LNode;
if (n == NULL)
    return false;
n->data = value;
n->prev = last;
if (last)
    last->next = n;
else
    first = n;
last = n;
size++;
return true;
}

11. Modify the implementations of InsertFirst and InsertLast from the previous problem so that they do not cause Segmentation faults.

12. What needs to be passed to the Forward and Backward functions defined and implemented in Lab 11?