EECS 560
Lab 1

Luke Dercher
TA: Apoorv Nishikant Ingle

1. The algorithmic complexity of the following operations is as follows.
    a. Adding an element at the end of the list: The complexity of this is O(1). This is because no matter what size the list is, the operation will always take the same amount of time equal to some constant. This is due to the fact that I'm checking whether the user is adding a node where the m_back pointer is. This eliminates the need to traverse the list.
    b. Find a given element: This is done in O(n) time. This is due to the fact that I have to traverse through the list to find the element whether it is in the list or not.
    c. Size of the list: This can be found in O(1). The reason for this is that the size of the list of a member variable of the class that is updated as nodes are added and removed.

2. What would be the optimal complexity of the operations described in 1a, 1b, and 1c?
    a. In 1a the optimal complexity is met. This is O(1).
    b. In 1b the optimal complexity is also met. This is O(n). This is because there in no other way to find an element in the list without traversing it.
    c. In 1c the optimal complexity is met. This is O(n).

3. What design decisions will you make to ensure you can get the most efficient algorithmic complexity
   for each of the operations that you analyzed for 1a, 1b and 1c?
    a. For 1a I would just use the fact that the location of the last node is already stored in memory "m_back". Therefore the process of finding the last position in the list is already done.
    b. For 1b it's not possible to find an element in the list without traversing the list. Therefor I would traverse the list to find the node.
    c. For 1c the size of the list is a member variable of the class "m_size" therefore it can be easily passed to the method.