CSC 326 Data Structures

More on Linked Lists
In some applications, it is convenient to keep access to both the first node and the last node in the list.

Sometimes a head node is used so that every node has a predecessor, which thus eliminates special cases for inserting and deleting.

The data part of the head node might be used to store some information about the list, as number of values in the list.
Sometimes a **trailer node** is also used so that **every node has a successor**.

Two or more lists can share the same trailer node.
In other applications (e.g., linked queues), a circular linked list is used; instead of the last node containing a NULL pointer, it contains a pointer to the first node in the list. For such lists, one can use a single pointer to the last node in the list, because then one has direct access to it and "almost-direct" access to the first node.
All of these lists, however, are uni-directional; we can only move from one node to the next. In many applications, bidirectional movement is necessary. In this case, each node has two pointers — one to its successor (null if there is none) and one to its predecessor (null if there is none.) Such a list is commonly called a **doubly-linked** (or **symmetrically-linked**) list.
We could modify this doubly-linked list so that both lists are circular forming a **doubly-linked ring**.
The STL list Class Template list is a sequential container that is optimized for insertion and erasure at arbitrary points in the sequence.

1. Implementation

As a circular doubly-linked list with head node. Its node structure is:

```c++
struct list_node
{
    pointer next, prev;
    T data;
}
```