Linked Lists

- Sequence of items, each item connected to the next by a link

- **Node = data value + link to next item**
- **NODE *headPtr** in LINKED_LIST points to first NODE in the list

- Linked list functionality:
    - clear
    - removeNode, removeHeadNode
    - findNodeByPosition
    - insertNodeAtHead

- What is big-O for these functions with linked list vs. **static** array? …vs. **dynamic** array?

Simple Pointer-Based Implementation

```
void LINKED_LIST::insertAtHead(const int newData) {
NODE *ptrToNewNode = new NODE(newData, NULL);

  ptrToNewNode->setLink(headPtr);
  headPtr = ptrToNewNode;
}

void LINKED_LIST::insert(const int newData, NODE* ptrToPrevNode) {
  if (ptrToPrevNode != NULL) {
    NODE *ptrToNewNode = new NODE(newData, NULL);
    ptrToNewNode->setLink(ptrToPrevNode->getLink( ));
    ptrToPrevNode->setLink(ptrToNewNode);
  }
  else insertHeadNode(newData); // assume we're inserting at head
}
```

What if our application needed to always insert at the 'tail' end of the list???

To be efficient, also maintain **NODE *tailPtr** for the LINKED_LIST

What needs to be added to other 'remove' and insert' functions???