# CS 445

# Rec 7

# Agenda

1. Topic of This Week
    a. Circular Doubly Linked List
2. Working Session: Lab 7

# Topic of the Week

## Circular Doubly Linked List

- Nodes contain <data, prev, next>
- Make use of the constructor `CDLL_Node(T data)` when creating the very first node

## Understand

- How to populate a circular doubly linked list
  - insertAtFront(), insertAtTail()
- How to traverse a circular doubly linked list
  - size(), search(), contains(), toString()

```
// inner class
// private to code outside of the file
// but public to code inside
class CDLL_Node<T>
{
  T data;
  CDLL_Node<T> prev, next;

  CDLL_Node()
  {
    this(null, null, null);
  }

  CDLL_Node(T data)
  {
    this(data, null, null);
  }

  CDLL_Node(T data, CDLL_Node<T> prev, CDLL_Node<T> next)
  {
    this.data = data;
    this.prev = prev;
    this.next = next;
  }

  // toString() must be public
  public String toString()
  {
    return "" + data;
  }

} //end Node class
```

# Illustration of `insertAtFront()`

```
public void insertAtFront(T data)
{
        CDLL_Node<T> newNode = new CDLL_Node<T>(data,null,null);
        if (head==null)
        {
                newNode.next=newNode;
                newNode.prev=newNode;
                head = newNode;
                return;
        }

        CDLL_Node<T> old1st=head,oldlast=head.prev;
        head=newNode;
        newNode.next=old1st;     //       E -> A
        newNode.prev=oldlast;   // D <- E
        old1st.prev=newNode;    //  E <- A
        oldlast.next=newNode;  //  D -> E
}
```

- **Understand base case**
- **Understand insertion**
- **How to reuse this code for `insertAtTail()`?**

# Lab 7

1. **insertAtFront()**: as you write it print the data of each node as you insert it. This will verify you are not hung in an infinite loop.
2. **size()**: once you are sure your insertAtFront() is at least creating new nodes and not infinite looping, traverse the list CLOCKWISE (following the next pointer ) and count the nodes.
3. **toString()**: once your size works, your InsertAtFront() is probably correct so now you should write your toString using the size() code as a template. Instead of incrementing a counter as the visitation operation, you tack that Node's data onto your string. (SEE OUTPUT SCREENSHOT FOR FORMATTING).
4. **search()**: it's just the Same traversal as size() but a different visitation operation ( .equals() ).
5. **contains()**: just return the success/failure of search()
6. **insertAtTail()**: use your insertAtFront as a starting point.