CSE 12 HW5 --- Circular Linked List

General Hints:

- Make use of `getPre()` `getNext()` `setPre()` `setNext()` of `Node`.

What To Write

ListEngine

- ListEngine Ctor
    - Correctly allocate each data field of `listEngine`.
    - Increment counter
    - `this`
- JettisonList:
    - Iterate through each node of the list to jettison each Node
- advanceNext
    - Move the `end` to the node of its next
    - Error Checking
- advancePre
    - Just like Advance Next
- isEmpty
    - Check occupancy
- insert
    - Check whether you are inserting the first node into a list (make use of `isEmpty`)
        - If it is the first node, then just allocate a node manually and set the `end` to be the new node you created.

          ```
          end = Node(...);
          ```

        - Otherwise
            - You call the `insert` of `node` on the `end`, set `end` to be the newly returned value
                - If it is inserting at the end, then you are done. Otherwise, you need to reset your `end`pointer back to its original value.
                - Therefore, you may still want to use your old end, so it would be better if you store it before assigning. So one possibility of implementation is:

```
// THIS IS PSEUDOCODE
Node tempEnd = end;
locate(...);
// nodeToInsert.insert(...);
end = end.insert(...);
if (condition) {
  end = tempEnd;
}
```

- *Note:*
  - You should call `insert` of Node only once to get full credit
  - You don't need to increment occupancy because it is incremented in the constructor of Node.

- checkToGoForward
  - Check to see which way to go.
    - When you have a list of length 1000 and you want to insert at the front, you probably want to go in the direction of going forward from your end to get to your front
    - When you have a list of length 1000 and you want to insert at the 999th position, you probably want to go in the direction of going backward from your end to get there.
    - If you are inserting right at the middle, you can go either way.
- locate
  - If you are locating at the End you have nothing to do
  - Else check to go forward to see which way to go
  - You can use different ways to implement it, you can even change the signature of the method.
    - By different ways, we mean how your `locate` is going to help `insert`.
    - One way is to return the place where `insert` should `insert` after. (Need change the function declaration)
    - Another way is to move the `end` to the correct place by calling advancePre vs advanceNext.
      - If you move the `end`, before calling `locate`, you need to store the original end just like what we did above in `insert`.
- remove
  - Very similar to insert. And you should only call `remove` of Node only once.
  - Error checking.
- view
  - Same as remove just don't remove anything
  - Error checking.
- writeReverseList

    - Refer to the writeList, just reverse it.
    - Should be a do-while ways of thinking instead of while. Or for-loop but move once first.
        - The reason is you want to start at the end not front.

Node Engine

- jettisonNodeAndData
    - You should jettison everything in the nodeEngine
        - Including data
- jettisonNodeOnly
    - Only jettison the nodeEngine
    - Because Data will be jettisoned in Driver.
    - You don't want to jettison it yet because you will return this data to display.
- insert
    - You should create a new `Node` to hold the a new `NodeEngine` to hold the data that passed in.
    - The `Node` that we created for you should be returned.
    - You need to attach & integrate the newNode into the list. Refer to your notes.
- remove
    - Very similar to insert, but you reverse the logic to detach the node to remove.
    - You should call `jettisonNodeOnly` because you want to return the data that will be removed.
- view
    - Remove without removing

Notes:

- This PA is emphasizing the idea of delegating
    - When you are writing code in `ListEngine`, just focus on what it should do, and don't think about `NodeEngine`.
        - For example, when `insert`, you `ListEngine` code need to focus on where to insert, but not about how to `insert`.
        - You `NodeEngine` will then deal with the process of attaching and integrating.
- Make use of the dev guide. It should be easy to follow.
- `set list`

This PA is pretty complicated, so Start Early!!!!