Project 5:
git + GitHub 
+
Recursive List Manipulation

For this project you will modify a **singly-linked list** class by adding a recursive method that will **invert** the order of the list and another one that will **rotate** the list. Both methods **must be recursive** and should not involve loops.

Moreover, you will **work with git version control**.

Thus in **Part 1** of the project you will:

- Set up a GitHub account if you don't have one.
- Learn or brush-up on git and GitHub
- Accept a GitHub classroom assignment
- Clone a repository with starter files for this project.

In **Part 2** of the project you will:

- Implement and test `LinkedList.invert()`

In **Part 3** of the project you will:

- Implement and test `LinkedList.rotate()`

Part 1 - getting started with GitHub Classroom:

- If you don't already have one, got to https://github.com/ and create a GitHub account.
- Next, watch this video to brush-up on or learn the basics of git and GitHub:

  https://www.youtube.com/watch?v=MJUJ4wbFm_A

- For this project we will use GitHub Classroom. The following video will guide you through the entire process: from accepting an assignment to submitting your solution.

  You will find the link for our GitHub Classroom on Blackboard.

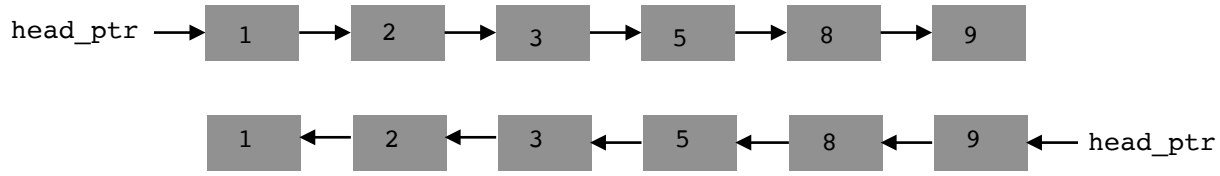
  Although the video is about a different course, the instructions are the same (with different repo and file names). The only difference is that we will not add a distribution branch, so you can **ignore** the part where it says to execute the two git commands in the readme file (there are not extra instructions in the readme file on our repo).

  This video will also show you how to submit to Gradescope via GitHub. Make sure to refer back to these instructions when it's time to submit.

  https://www.youtube.com/watch?v=AHDCokfgcSo

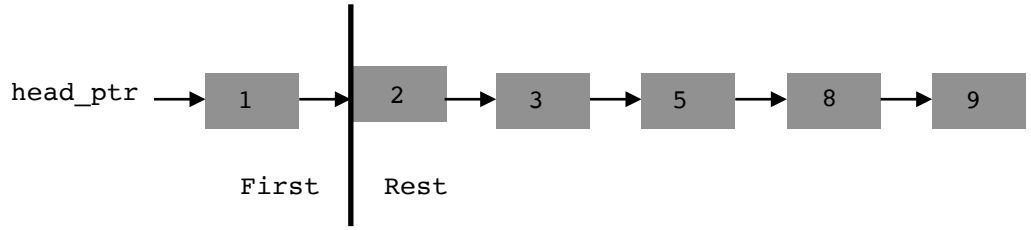
Part 2 - Recursive `LinkedList::invert()`:

The idea: In a linked list it is possible to invert the order of its elements in **O(n) time** and **in place (O(1) extra space)** by simply changing the direction of the links:

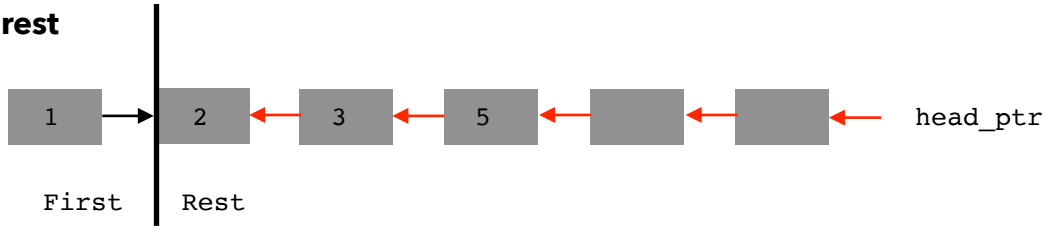


To do so recursively, you can think of the solution to the problem as:

- **Split the list into first and rest**



- **Invert the rest**



- **Reconnect rest to first in the appropriate direction**

Implementation:

Modify the LinkedList class as follows:
In order to practice safe programming and not expose the structure of the list to clients of the class via pointer parameters, you will write a **public non-recursive method `invert`** that calls a **private recursive method `invertRest`.**
You must implement these methods in O(n) time and **in place (you cannot use any additional list or other containers and cannot make copies of the nodes**)
Should you write a solution that uses iteration instead of recursion and /or uses additional containers to invert the list, I reserve the right to take away points even if Gradescope gives you full credit.

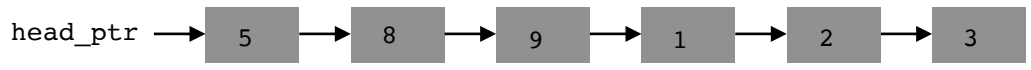
```
// A  wrapper to a recursive method that inverts the contents of the list
// @post the contents of the list are inverted such that
//       the item previously at position 1 is at position item_count_,
//       the item previously at position 2 is at position item_count_-1 ...
//       the item previously at position [item_count/2] is at position
         [item_count_/2]
void invert();

//private function to invert, used for safe programming to avoid
//exposing pointers to list in public methods
// @post the contents of the list are inverted such that
//       the item previously at position 1 is at position item_count_,
//       the item previously at position 2 is at position item_count_-1 ...
//       the item previously at position [item_count/2] is at position
        [item_count_/2]
void invertRest(Node<T>* current_first_ptr);
```

PART 3 - Recursive `LinkedList::rotate()`:

The idea: Given a linked list, rotating by k means to shift all items k positions to the right with wrap around at the end. Thus, given the list:

`rotate(3)` will produce:



It is possible to rotate a list's elements **in place (O(1) extra space)** by relinking the nodes currently in the list. **You must do so recursively!!!**
Hint: think of the problem as "perform the smallest rotation (i.e. k = 1) by moving the last node to be the first and then rotate the rest recursively (k-1)".

Implementation:

```
// @pre k >= 0
// @post the contents of the list are rotated to the right by k places, s.t.
//       every element at position i shifts to position i+k % item_count_
void rotate(int k);
```

Note: rotate is a public function and does not need a private helper because it does not take pointer parameters.

Submission: refer back to video

You will find instructions on how to submit to Gradescope through GitHub at the end of the GitHub Classroom tutorial video (same link provided again here)

- https://www.youtube.com/watch?v=AHDCokfgcSo

The due date is Friday November 1 by 5pm. No late submissions will be accepted.

Have Fun!!!!!