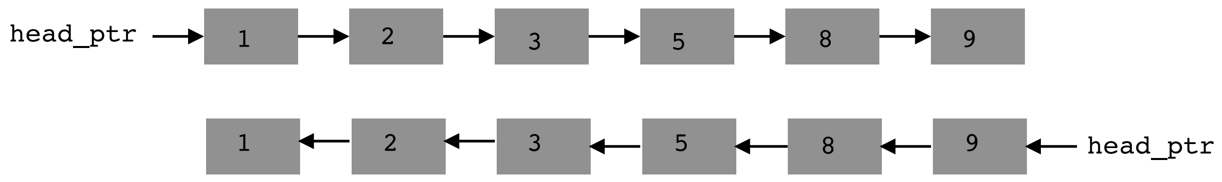
Project 4: Recursive Invert

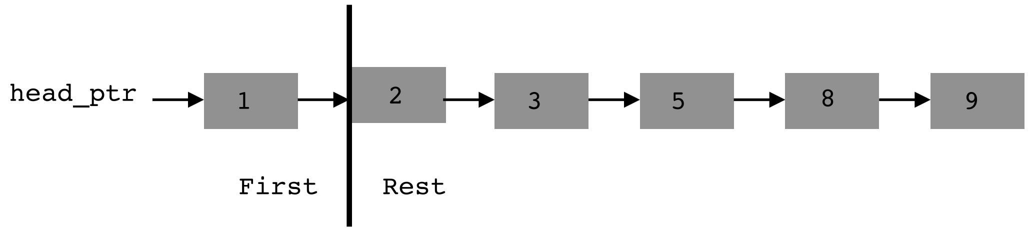
For this project you will modify a **singly-linked list** class by adding a method that will invert the order of the list. This method **must be recursive** and should not involve loops.

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)** simply by changing the direction of the links:

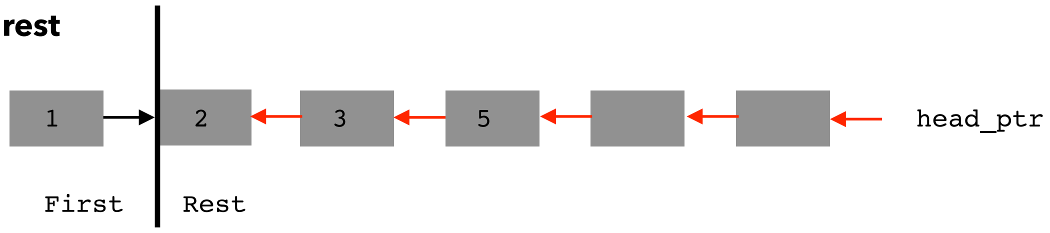


To do so recursively, you can think of the solution to the problem as:

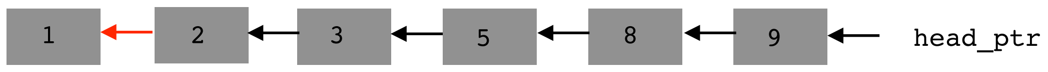
- **Split the list into first and rest**



- **Invert the rest**



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



Implementation:

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);
```

Testing:

In `main()` (not for submission) instantiate a `LinkedList` object and add data (nodes) to it. The list could be of any type and it should not affect execution. Once you have data in the list, call invert and make sure it works correctly (e.g. display the contents of the list to check that the order is the opposite in which you inserted the data items).
Also test that your methods run correctly on edge cases (e.g. empty list, single-node list).

Grading Rubric:

- **Correctness 100%** (distributed across unit testing of your submission)

Notes:

- I reserve the right to detract points given by Gradescope if your submission does not comply in some way with this specification.
- A submission that implements all required classes and/or functions but does not compile will receive 40 points total (including documentation and design).

Submission:

You will find the Node and LinkedList class on Blackboard under Course Materials/Project4. You must modify and **submit only `LinkedList.cpp` (1 file)**

Your project must be submitted on Gradescope.
Although Gradescope allows multiple submissions, it is not a platform for testing and/or debugging and it should not be used for that. You MUST test and debug your program locally.
Before submitting to Gradescope you MUST ensure that your program compiles (with g++) and runs correctly on the Linux machines in the labs at Hunter (see detailed instructions on how to upload, compile and run your files in the "Programming Rules" document). That is your baseline, if it runs correctly there it will run correctly on Gradescope and if it does not, you will have the necessary feedback (compiler error messages, debugger or program output) to guide you in debugging, which you don't have through Gradescope.
"But it ran on my machine" is not a valid excuse to get credit.
Once you have done all the above you submit it to Gradescope.

The due date is Thursday June 27 by 9pm. No late submissions will be accepted.

Have Fun!!!!!