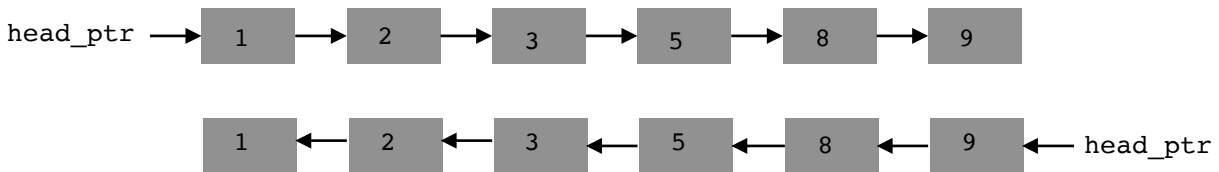


Project 3A: Recursive Invert

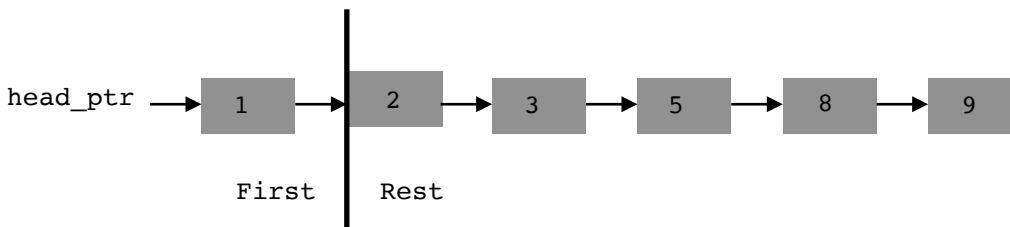
For this project you will modify a **singly-linked list** class by adding a method that will invert the order of the list. This method **must be recursive** and should not involve loops.

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:

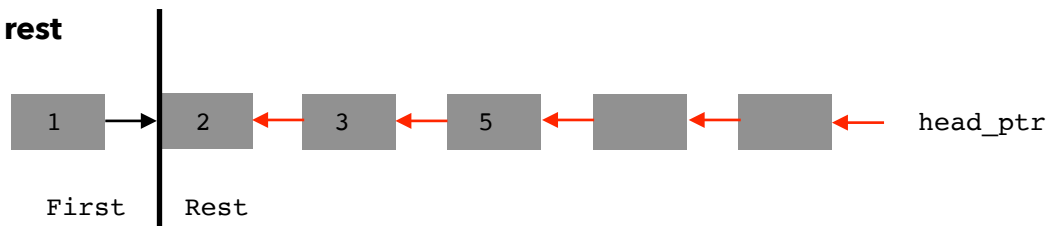


To do so recursively, you can think of the solution to the problem as:

- **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. Moreover, if your name is not in your submission's comment preamble you will lose 10 points.

```
// 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);
```

Project Notes: The List class will issue some **warnings** for `getEntry()`, because an uninitialized item is returned if the function is called with an invalid position. This is so because we have not discussed exception handling yet. We will be able to fix this later in the semester.

Submission:

You will find the Node and LinkedList class on Blackboard under Course Materials/Project3. You must modify and **submit only LinkedList.cpp (1 file)**

Your project must be submitted on Gradescope. The due date is Friday April 5 by 6pm. No late submissions will be accepted.

Have Fun!!!!