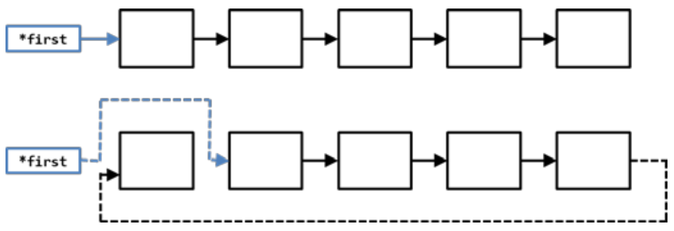
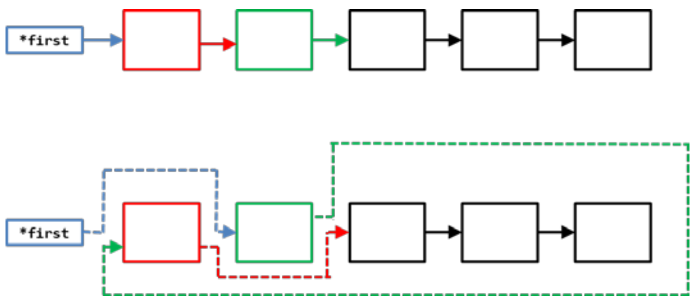
Singly- and Doubly-linked Lists: Practice Suggestions

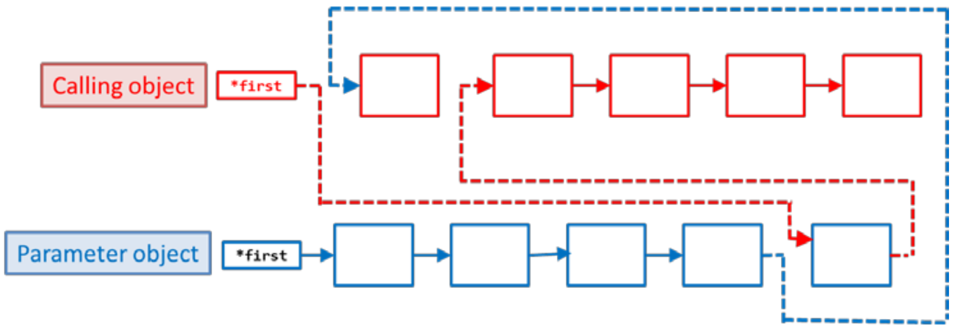
These are examples of functions that can be implemented to practice linked lists. All these functions apply to both **SLL** and **DLL** and should be implemented as **member** functions of the respective classes.

Details to keep in mind:

- You will need to determine whether the **function** is **const** and, in the case you are passing parameters, whether the **parameter**(s) should be passed by **reference** and as **const**.
- You may or may not assume that objects are empty.
- When changing the position of nodes within the list, it is important that you check that all pointers in a node are set properly. For example, if you are deleting the first node of a doubly-linked list, you need to make sure that the previous pointer of the new first node is pointing to NULL. The best way to keep track of all pointers is by drawing the list.

You can find additional ideas at **codingbat.com**. You simply need to modify the functions so that they can work in a linked list, and you can use of all the testing cases available for the functions.

No parameters	**Re-arrange the list** by re-setting pointers (**NOT** values) • Any combination of re-arranging the list by moving any node to a different position. You are not actually "moving" the nodes in memory, but you are re-arranging the way **pointers** are pointing. • **Example:** Move the **first node** to the **end of the list**: 
	Swap nodes by re-setting pointers (not values). • Any combination of swapping any two nodes in the list. You are not actually "moving" the nodes, but you are re-arranging the way **pointers** are pointing. • **Example:** Swap the **first node** with the **second node**: 
	Delete nodes • Delete a node in the list. • **Example:** Function **deleteFirst** – Delete the **first** node in the list.

Parameter: A list object	**Swap nodes** between **two objects** by re-setting **pointers** (not values). • Any combination of swapping a node in the calling object with a node in the parameter object. You are not actually “moving” the nodes, but you are re-arranging the way pointers are pointing. • **Example:** Given a linked list passed as a parameter, swap the **first** node of the calling object with the **last** node of the parameter object. 
	Copying <u>values</u> from one list to another. • **Example:** Given a linked list passed as a parameter, copy all values stored in the calling object to the end of the parameter object. You will need to add new nodes to the parameter object. Calling object is: 1 2 3 4 5 6 Parameter object is: 10 11 12 Parameter object becomes: 10 11 12 1 2 3 4 5 6 There are several possible combinations: • Copy in reverse • Copy only specific data (even numbers, odd numbers, etc.)
Parameters: A list object An array	**Copying** values, given three different containers. • **Example:** Given two parameters, a linked list and an array, copy all values stored in the parameter object to the beginning of the calling object, and copy all values stored in the array to the end of the calling object. You will need to add new nodes to the calling object. Calling object is: 1 2 3 4 5 6 Parameter object is: 10 11 12 Array is: 30 31 32 Calling object becomes: 1 2 3 4 5 6 10 11 12 30 31 32 There are several possible combinations: • Copy in reverse • Copy only specific data (even numbers, odd numbers, etc.)