Singly Linked List Abstract Data Structure Contract

Properties

- **head**: A reference to the first Node in the list. Initialized to `None`.
- **tail**: A reference to the last Node in the list. Allows for $O(1)$ insertions at the end.
- **length**: An integer tracking the total number of nodes currently in the list.

Methods

1. Initialization and Conversion

- **init(data_list=None)**:
  - Constructor for the Linked List.
  - If an iterable (like a Python list) is provided, the object should automatically populate itself with nodes containing those values.
- **to_list()**:
  - Traverses the entire linked list and returns a standard Python list of all node values in order.
- **repr()**:
  - Returns a string representation of the list, typically formatted to show the direction of links (e.g., “val1 -> val2 -> None”).

2. Insertion Operations

- **prepend(value)**:

    - Adds a new node with the given value to the very start of the list.
    - Updates the **head** and, if the list was empty, the **tail**.
- **append(value)**:
    - Adds a new node with the given value to the very end of the list.
    - Uses the **tail** property for $O(1)$ efficiency.
- **insert(index, value)**:
    - Adds a new node at a specific zero-indexed position.
    - Requires shifting existing nodes and updating the relevant pointers.

3. Deletion Operations

- **pop_first()**:
    - Removes the first node from the list and returns its value.
    - Updates the **head** to the next node in the sequence.
- **pop()**:
    - Removes the last node from the list and returns its value.
    - Requires a traversal to the penultimate node to update the **tail** reference.
- **remove_at(index)**:
    - Removes the node at the specified index.
    - Re-links the preceding node directly to the succeeding node.

4. Search and Utility

- **get(index)**:
    - Returns the value of the node at the specified index without removing it.
- **find(value)**:
    - Searches for a value and returns the index of its first occurrence.
    - Returns `None` or raises an error if the value does not exist.
- **clear()**:
    - Empties the list by resetting **head**, **tail**, and **length**.

Constraints and Complexity Goals

- **Size Lookup**: Must be $O(1)$ by utilizing the **length** property.
- **Prepend/Append**: Must be $O(1)$ by utilizing **head** and **tail** pointers.
- **Search/Random Access**: $O(n)$ average and worst-case time complexity.