Design your implementation of the linked list. You can choose to use the singly linked list or the doubly linked list. A node in a singly linked list should have two attributes: val and next. val is the value of the current node, and next is a pointer/reference to the next node. If you want to use the doubly linked list, you will need one more attribute prev to indicate the previous node in the linked list. Assume all nodes in the linked list are 0-indexed.

Implement these functions in your linked list class:

* get(index) : Get the value of the index-th node in the linked list. If the index is invalid, return -1.
* addAtHead(val) : Add a node of value val before the first element of the linked list. After the insertion, the new node will be the first node of the linked list.
* addAtTail(val) : Append a node of value val to the last element of the linked list.
* addAtIndex(index, val) : Add a node of value val before the index-th node in the linked list. If index equals to the length of linked list, the node will be appended to the end of linked list. If index is greater than the length, the node will not be inserted.
* deleteAtIndex(index) : Delete the index-th node in the linked list, if the index is valid.

**Example:**

**Input:**

["MyLinkedList","addAtHead","addAtTail","addAtIndex","get","deleteAtIndex","get"]

[[],[1],[3],[1,2],[1],[1],[1]]

**Output:**

[null,null,null,null,2,null,3]

**Explanation:**

MyLinkedList linkedList = new MyLinkedList(); // Initialize empty LinkedList

linkedList.addAtHead(1);

linkedList.addAtTail(3);

linkedList.addAtIndex(1, 2); // linked list becomes 1->2->3

linkedList.get(1); // returns 2

linkedList.deleteAtIndex(1); // now the linked list is 1->3

linkedList.get(1);    // returns 3

**Constraints:**

* 0 <= index,val <= 1000
* Please do not use the built-in LinkedList library.
* At most 2000 calls will be made to get, addAtHead, addAtTail,  addAtIndex and deleteAtIndex.