To implement a stack using a linked list, you can use a singly linked list:

* The top of the stack will be the head of the linked list.
* All stack operations (push, pop, getTop, and isEmpty) will operate on the head of the linked list, ensuring that each operation runs in O(1) time.
  + We must define the Node Structure:
    - Each node in the linked list will store:
      * The data (value of the stack element).
      * A next pointer to the next node in the list.
  + Define the Stack Class:
    - The stack will maintain a pointer to the head of the linked list.
  + Implement the push operation:
    - Create a new node with the given data.
    - Set the next pointer of the new node to the current top.
    - Update top to point to the new node.
  + Implement the pop operation:
    - Check if the stack is empty (i.e., top is None).
    - If not empty, store the value of the top node.
    - Update top to point to the next node.
    - Return the stored value.
  + Implement the getTop operation:
    - Check if the stack is empty (i.e., top is None).
    - If not empty, return the stored value.
  + Implement isEmpty:
    - Check if top is None.

This solution has:

* Time Complexity:
  + push: O(1) ;
  + pop: O(1);
  + **getTop:** O(1);
  + isEmpty: O(1);

This solution is simple to implement and leverages the natural behavior of a linked list.

To implement a QUEUE using a linked list, you can use a singly linked list and follow these steps:

* We must define the Node Structure:
  + Each node in the linked list will store:
    - The data (value of the stack element).
    - A next pointer to the next node in the list.
* Define the Queue Class:
  + The queue will maintain **two main pointers, front and tail, to keep track of the front and tail of the queue.**
  + The class will maintain a size variable in the Queue class.
* Implement the Dequeue operation:
  + Check if the queue is empty (i.e., front and tail pointer as None).
  + If not empty:
    - Retrieve the value from the front node.
    - Update the front pointer to the next node.
    - Return the value stored.
    - If the queue becomes empty, set the tail and font pointers to null.
    - Adjust the size variable.
* Implement the Enqueue operation:
  + Create a new Node.
  + If the queue is empty, update both front and tail pointers to the new node.
  + If not, set the next pointer of the current tail to the new node and update the tail pointer to the new node.
    - Adjust the size variable.
* Implement Size operation:
  + Read the size variable.
* Implement isEmpty operation:
  + Verify if both front and tail pointer are None.

This solution has:

* Time Complexity:
  + ENQUEUE: O(1) ;
  + DEQUEUE: O(1);
  + **SIZE:** O(1);
  + isEmpty: O(1);