A01641147 – Carlos Alberto Veryan Peña
Documentation Act 2.1 - Linear data structure ADT execution

## CREATE Function

Beginning

1. We create a new node and assign it to a pointer. We can also do it in one line: Node *new_node = new Node(val);
2. We initialize the new node. We have to set the previous and next pointers to NULL.
3. If the list is not empty (head != NULL) we have to set the previous pointer of the first node to the new node and the next pointer of the new node to the first node.
4. Finally we return the new node (the new head of the list).

End

1. We have 2 pointers which are head and temp. Head points to the first node in the list and temp points to the last node in the list.
2. We have a while loop which runs till the temp node is not null.
3. Inside the while loop, we update the temp to point to the next node in the list.
4. After the while loop is over, we create a new node, allocate memory for it and store the value in the new node.
5. We update the next pointer of the last node to point to the newly created node.
6. We update the previous pointer of the newly created node to point to the last node.
7. We return the head pointer which points to the first node in the list.

The time complexity of this algorithm is O(n) because we are traversing through the entire linked list to find the last node.

In a Specific Position

1. Create a new node
2. Assign the value to the new node
3. Check if the list is empty
4. Traverse the list until the node with the value after is found
5. Assign the new node to the next pointer of the node with the value after
6. Assign the next pointer of the node with the value after to the new node
7. Assign the new node to the previous pointer of the node after the node with the value after
8. Check if there is a node after the node with the value after

9. Assign the new node to the previous pointer of the node after the node with the value after.

## READ Function.

1. We create a temporary pointer and set it to point to the head of the list.
2. We loop through the list until we find the value or until we reach the end of the list.
3. If we find the value, we return the index.
4. If we don't find the value, we return -1.

## UPDATE Function.

1. We create a pointer temp and point it to the head of the linked list
2. We then create a while loop to traverse the linked list until the end
3. In the while loop, we first check if the value of the current node is equal to the value we want to replace
4. If so, we replace the value of the current node with the value we want to replace it with and return the head
5. If not, we move on to the next node and repeat the process
6. If we reach the end of the linked list, we return the head.

## DELETE Function.

1. If the head is NULL, then return the head, since there is nothing to delete.
2. If the head is the node we want to delete, then return the head->next, since we don't need to keep the head.
3. If the head is not the node we want to delete, then save the head, and delete the rest of the linked list.