Tutorial – Linked Lists

1. **(moveEvenItemsToBackLL)** Write a Python function move_even_items_to_back_ll() that moves all the even integers to the back of the linked list.

   The function prototype is given as follows:

   ```
   def move_even_items_to_back(ll: LinkedList):
   ```

   Some sample inputs and outputs sessions are given below:

   If the linked list is **2, 3, 4, 7, 15, 18**:

   ```
   The resulting Linked List after moving even integers to the back of
   the Linked List is: 3 7 15 2 4 18
   ```

   If the linked list is **2, 7, 18, 3, 4, 15**:

   ```
   The resulting Linked List after moving even integers to the back of
   the Linked List is: 7 3 15 2 18 4
   ```

   If the current linked list is **1, 3, 5**:

   ```
   The resulting Linked List after moving even integers to the back of
   the Linked List is: 1 3 5

   If the current linked list is 2 4 6:
   The resulting Linked List after moving even integers to the back of
   the Linked List is: 2 4 6
   ```

2. **(moveMaxToFront)** Write a Python function move_max_to_front() that traverses a linked list of integers at most once, then moves the node with the largest stored value to the front of the list.

   The function prototype is given as follows:

   ```
   def move_max_to_front(ll: LinkedList):
   ```

   For example, if the linked list is (**30, 20, 40, 70, 50**), the resulting linked list will be (**70, 30, 20, 40, 50**).

   ```
   1: Insert an integer to the linked list:
   2: Move the node with the largest stored value to the front of the
   list:
   0: Quit:

   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 30
   The Linked List is: 30
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 20
   The Linked List is: 30 20
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 40
   The Linked List is: 30 20 40
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 70
   The Linked List is: 30 20 40 70
   ```

```
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 50
The Linked List is: 30 20 40 70 50

Please input your choice(1/2/0): 2
The resulting Linked List is: 70 30 20 40 50

Please input your choice(1/2/0): 0
```

3. **(removeDuplicatesSortedLL)** Write a Python function `remove_duplicates_sorted_ll()` that removes all duplicate values from a sorted linked list. You may assume that the list is already in ascending sorted order**.**

   The function prototype is given below:

   ```
   def remove_duplicates_sorted_ll(ll: LinkedList):
   ```

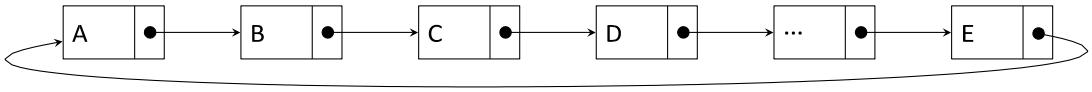
   For example:
   If the linked list is (1, 2, 2, 4, 4, 5, 5), the resulting linked list will be (1, 2, 4, 5).
   If the linked list is (1, 2, 3, 4, 5), the resulting linked list will be (1, 2, 3, 4, 5)

   Sample test cases are given below:

   ```
   1: Insert an integer to the linked list:
   2: Remove duplicates from a sorted linked list:
   0: Quit:

   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 1
   The resulting linked list is: 1
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 2
   The resulting linked list is: 1 2
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 2
   The resulting linked list is: 1 2 2
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 4
   The resulting linked list is: 1 2 2 4
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 4
   The resulting linked list is: 1 2 2 4 4
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 5
   The resulting linked list is: 1 2 2 4 4 5
   Please input your choice(1/2/0): 1
   Input an integer that you want to add to the linked list: 5
   The resulting linked list is: 1 2 2 4 4 5 5

   Please input your choice(1/2/0): 2
   The resulting linked list after removing duplicate values from the
   sorted linked list is: 1 2 4 5

   Please input your choice(1/2/0): 0
   ```

4. We assign the link of the last node to the first node instead of assigning it to a null value. This turns the linked list into a circular linked list. Let `Aptr` and `Bptr` point to any two nodes in the linked list. What is the outcome of the following functions?

Figure 1.1: A Circular Linked List




```
def T2Q1 (s,q):
     temp = s
     while temp.next != q:
          temp = temp.next
     temp.next = s

T2Q1 (Aptr, Bptr)
T2Q1 (Bptr, Aptr)
```