Tutorial 1 (Week 4)
Linked Lists

Instructor: Dr. Quah T.S., Jon
Email: itsquah@ntu.edu.sg

Q1 (move even items to back)

Full program is uploaded to our tutorial folder

```
63 def moveEvenItemsToBack(ll):
64     if ll.size < 2:
65         return
66
67     cur = ll.head
68     index = 0
69     for _ in range(ll.size):
70         if cur.item % 2 == 0:
71             even_value = cur.item
72             cur = cur.next
73             ll.removeNode(index)
74             ll.insertNode(ll.size, even_value)
75         else:
76             cur = cur.next
77             index += 1
78
```

Q2 (move max to front)

Full program is uploaded to our tutorial folder

```
72 def moveMaxToFront(ll):
73     if ll.head is None or ll.head.next is None:  # Empty list or single node
74         return -1
75
76     pre_max = None
77     max_node = ll.head
78     cur = ll.head
79
80     while cur.next is not None:
81         if cur.next.item > max_node.item:
82             pre_max = cur
83             max_node = cur.next
84         cur = cur.next
85
86     if pre_max is None:  # Max node is already at the front
87         return 0
```

Q2 (move max to front)(cont'd)

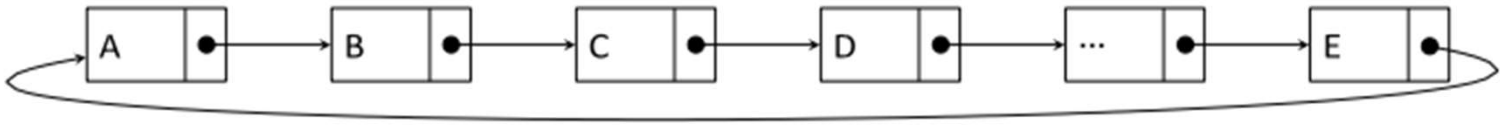
```
88
89      # Move the max node to the front
90      pre_max.next = max_node.next
91      max_node.next = ll.head
92      ll.head = max_node
93
94      return 0
```

Q3 (Remove duplicates)

Full program is uploaded to our tutorial folder

```
11 # Function to remove duplicates from a sorted linked
   list
12 def remove_duplicates_sorted_ll(ll):
13     current = ll.head
14     if current is None:
15         return
16
17     while current.next is not None:
18         if current.item == current.next.item:
19             temp = current.next.next
20             current.next = temp
21             ll.size -= 1
22         else:
23             current = current.next
```

Q4 (circular lists)



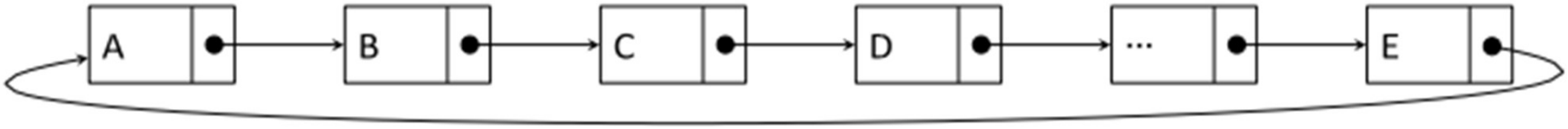
`T2Q1 (s,q):`

This function modifies the circular linked list. Here's what it does:

1. It starts from the node `s`.
2. Traverses the list until it finds the node that comes BEFORE node q
3. Updates the `next` pointer of this node to point back to `s`.

Let's use the example where Aptr points to B and Bptr points to D:

Q4 (circular lists) (cont'd)

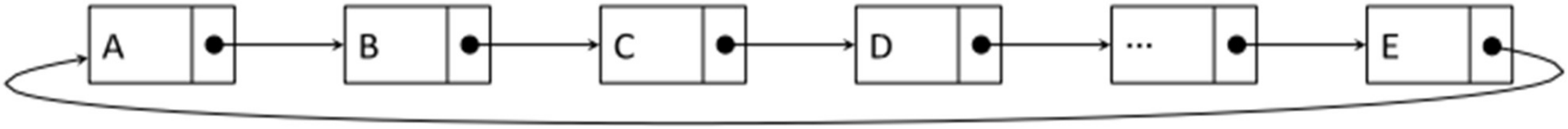


Let's use the example where Aptr points to B and Bptr points to D:

For `T2Q1(Aptr, Bptr):`

1. Start at B (s = Aptr = B)
2. Traverse until we find the node before D (which is C)
3. Make C point back to B
4. Result: B → C → B becomes one circle

Q4 (circular lists) (cont'd)



Let's use the example where Aptr points to B and Bptr points to D:

For `T2Q1(Bptr, Aptr):`

1. Start at D (s = Bptr = D)
2. Traverse until we find the node before B (which is A)
3. Make A point back to D
4. Result: D → E → A → D becomes another circle

- One circular list starts at `Aptr` and includes nodes up to (and including) the node BEFORE `Bptr`. This means B → C → B forms the first circle.
- Another circular list starts at `Bptr` and includes nodes up to (and including) the node BEFORE `Aptr`. This means D → E → A → D forms the second circle.