

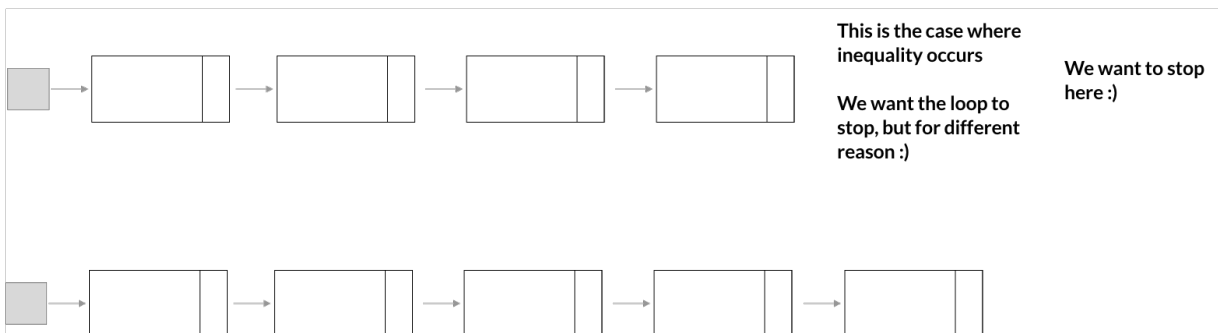
# CSC148 Worksheet 13 Solution

Hyungmo Gu

April 23, 2020

## Question 1

- a. The following diagram tells us the stopping condition occurs when both *curr1* and *curr2* is *None*.



Using this fact, the python expression involving *curr1* and *curr2* that expresses the stopping condition is

```
1 (curr1 is not None) and (curr2 is not None)
2
```

- b. Python expression for the while loop condition is

```
1 while (curr1 is not None) and (curr2 is not None):
2     ...
3
```

- c. The code for traversing two list is

```
1 while (curr1 is not None) and (curr2 is not None):
2     if curr1 is None or curr2 is None:
3         return False
4
5     if curr1.item != curr2.item:
6         return False
7
8     curr1 = curr1.next
9     curr2 = curr2.next
10
```

- d. After the loop ends, we know all items in `curr1` and `curr2` are identical.
- e. Because we know on successful loop termination, all items in `curr1` and `curr2` are the same, we can use this information to conclude the two linked lists have the same length.
- f. The code that should go after the end of while loop is

```

1     return True
2

```

## Question 2

- a. Initially, `curr` and `i` are as follows

```

1     curr = self._first
2     i = 0
3

```

- b. The stopping condition for the while loop is

```

1     curr is not None
2

```

Using this fact, we can conclude that the while loop condition is

```

1     while curr is not None:
2         ...
3

```

- c. The code for the loop body is

```

1     # 2. If index - 1 != current_index, then continue to next node
2     if index - 1 != current_index:
3         curr = curr.next
4         current_index += 1
5         continue
6
7     # 3. If curr.next is none, then let it terminate naturally
8     if curr.next is None:
9         curr = curr.next
10        current_index += 1
11        continue
12
13    # 4. If index - 1 == current_index, then return item of curr.next
14    return curr.next.item
15

```

- d. After the loop ends, we know `curr` is `None` and `index == len(self)`.

Using this fact, we can write that the post-loop code is

```

1      raise IndexError
2

```

```

1      def __getitem__(self, index: int) -> Any:
2          """Return the item at position <index> in this list.
3          Raise an IndexError if the <index> is out of bounds.
4          Precondition: index >= 0.
5
6          >>> lst = LinkedList([1, 2, 3])
7          >>> print(lst[0])
8          1
9          >>> print(lst[1])
10         2
11         >>> print(lst[2])
12         3
13         >>> print(lst[3])
14         Traceback (most recent call last):
15             ...
16         IndexError
17         """
18
19         curr = self._first
20         current_index = 0
21
22         # 1. If index == 0 and curr is not none, then return curr.item (
edge case)
23         if index == 0 and (curr is not None):
24             return curr.item
25
26         while curr is not None:
27             # 2. If index - 1 != current_index, then continue to next node
28             if index - 1 != current_index:
29                 curr = curr.next
30                 current_index += 1
31                 continue
32
33             # 3. If curr.next is none, then let it terminate naturally
34             if curr.next is None:
35                 curr = curr.next
36                 current_index += 1
37                 continue
38
39             # 4. If index - 1 == current_index, then return item of curr.
next
40             return curr.next.item
41
42         raise IndexError

```

Listing 1: worksheet\_13\_q2\_solution.py