Challenge 9: Union & Intersection of Linked Lists

In this lesson, linked lists meet data set operations.

We'll cover the following

Problem Statement

Union and **intersection** are two of the most popular operations which can be performed on data sets. Now, you will be implementing them for linked lists! Let's take a look at their definitions:

Union

> Given two lists, **A** and **B**, the union is the list that contains all objects that belong to either **A**, **B**, or to both.

Intersection

> Given two lists, **A** and **B**, the intersection is the largest list which contains all the elements that are common to both the sets.

The `union` function will take two linked lists and return their union.

The `intersection` function will return all the elements that are common between two linked lists.

Input

Two linked lists.

Output

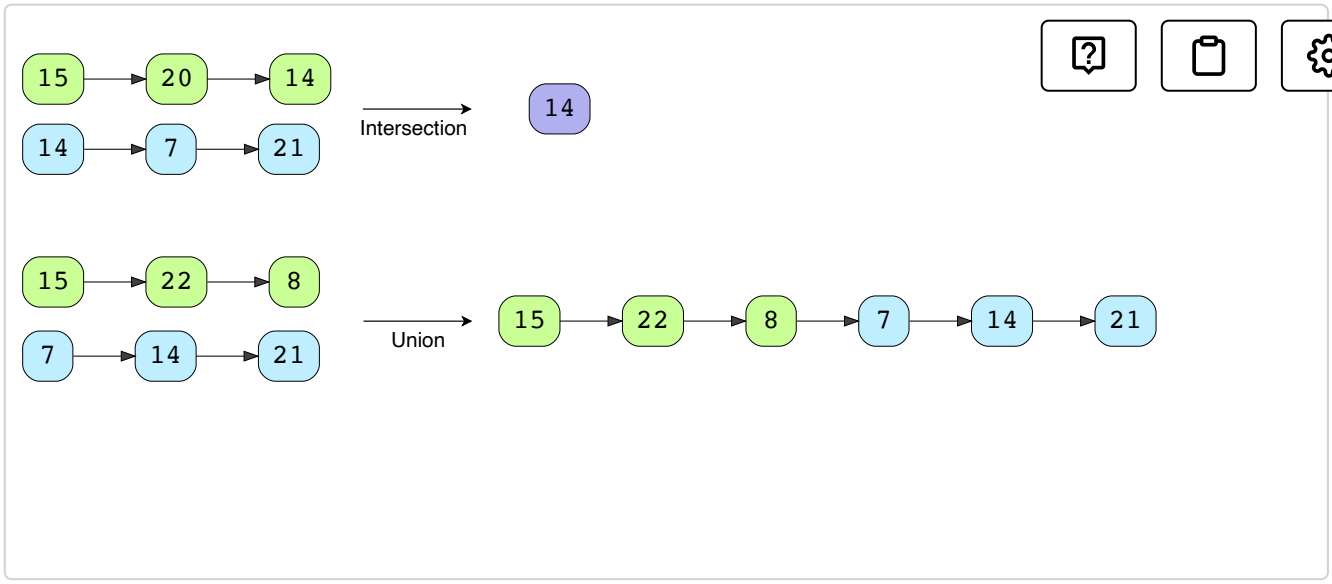
- A list containing the union of the two lists.
- A list containing the intersection of the two lists.

Sample Input

```
list1 = 10->20->80->60
list2 = 15->20->30->60->45
```

Sample Output

```
union = 10->20->80->60->15->30->45
intersection = 20->60
```



Coding Exercise

Design a step-by-step algorithm for the problem before jumping on to the implementation.

We are assuming that `union` and `intersection` will remove duplicates. For this reason, the `remove_duplicates` method has been provided to you as a member function of the `LinkedList` class.

If you get stuck, you can always refer to the solution provided in the solution section.

Good luck!

main.py

LinkedList.py

Node.py

```
from LinkedList import LinkedList
from Node import Node
# Access head_node => list.get_head()
# Check if list is empty => list.is_empty()
# Delete at head => list.delete_at_head()
```

```
# Delete by value => list.delete(value)
# Search for element => list.search()
# Length of the list => list.length()
# Remove duplicates => list.remove_duplicates()
# Node class  {int data ; Node next_element;}

# Returns a list containing the union of list1 and list2


def union(list1, list2):
    # Write your code here
    if list1.is_empty():
        return list2
    list1_curr = list1.get_head()
    while list1_curr.next_element:
        list1_curr = list1_curr.next_element
    if not list2.is_empty():
        list1_curr.next_element = list2.get_head()
    return list1#.remove_duplicates()

# Returns a list containing the intersection of list1 and list2


def intersection(list1, list2):
    # Write your code here
    temp = set()
    list1_curr = list1.get_head()
    while list1_curr:
        temp.add(list1_curr.data)
        list1_curr = list1_curr.next_element

    new_list = LinkedList()

    dummy = Node(0)
    dummy_head = dummy
    new_list.insert_at_head(dummy)
    list2_curr = list2.get_head()
    while list2_curr:
        if list2_curr.data in temp:
            dummy.next_element = Node(list2_curr.data)
            dummy = dummy.next_element
        list2_curr = list2_curr.next_element

    # print(temp)
    # return list2
    return new_list.get_head()
```

Back

Solution Review: Remove Duplicates f...

Next

Solution Review: Union & Intersection ...

Mark as Completed

Report an Issue