Week 4 worksheet: Collection Classes (Bag example)
Total points: 12 (graded out of 10)
Out: 2024 October 7 (Monday)
Due: 2024 October 9 (Wednesday end of day [2359 CDT according to D2L])
No late submissions will be accepted

What to submit?

Upload only one Word or PDF file to the designated D2L folder. Some questions may be answered directly on this file. Other questions can be answered on a piece of paper, and you take a picture of your paper and insert the picture in this file.

For this worksheet, I encourage you to write the code (preferably using your IDE of choice) to make sure it works, but do not submit your .java files. On the other hand, for each question in this worksheet, if the questions is asking for code writing, include a screenshot of the code that gives the answer to the question.

Exercise 1: how linked lists work?

[2.5 pts total] Suppose we have three classes, as shown below. Note that for simplicity, the instance variables are public so that we do not need getters and setters.

```
public class P {
      public int data;
      public Q next;
}
```

```
public class Q {
     public int data;
     public R next;
}
```

```
public class R {
     public int data;
     public P next;
}
```

Assume the following code is executed:

```
P p = new P();
Q q = new Q();
R r = new R();
p.data = 1;
q.data = 2;
r.data = 3;
```

If typing your answers, use → (- -> if you don't use Word) to show links.

a) [0.5 pt] Write code to make the "next" field of p point to q. Show the resulting linked list.

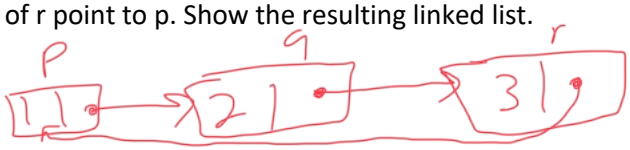
```
p.next = q;
```

b) [0.5 pt] Write code to make the "next" field of p point to q and the "next" field of q point to r. Show the resulting linked list.

```
p.next = q;
q.next = r;
```

c) [0.5 pt] Write code to make the" next" field of p point to q, the " next" field of q point to r, and the " next" field of r point to p. Show the resulting linked list.

```
p.next = q;
q.next = r;
r.next = p;
```



d) [1 pt] Suppose we have the following code executed.

```
p.next = q;
q.next = r;
r.next = p;
```

What is the output of each of the following statements?:

- `System.out.println(p.data);` 1
- `System.out.println(p.next.data);` 2
- `System.out.println(p.next.next.data);` 3
- `System.out.println(p.next.next.next.data);` 1
- `System.out.println(p.next.next.next.next.data);` 2

Exercise 2: using `class IntNode`

[5.5 pts] Consider the following piece of code that uses **IntNode**. Show the linked list with that started with the head node h after executing each line of code. Assume h is at position 1. Write your answer on the next page.

```
21          IntNode h = new IntNode(10,null);
22          h.addNodeAfter(30);
23          h = new IntNode(20,h);
24          h.addNodeAfter(40);
25          IntNode n = IntNode.listSearch(h, 10);
26          n.addNodeAfter(50);
27          h.addNodeAfter(60);
28          IntNode m= IntNode.listPosition(h, 5);
29          m.addNodeAfter(70);
30          m = h.getLink();
31          m.removeNodeAfter();
32
```

Write your answers in the following format (whether typed or legibly handwritten):

By the way, this is NOT the right answer to anything. Suppose the list is as follows, from head to tail:

- Node “h” with value 44
- Node “k” with value 33
- Node (anonymous) with value 88
- Node (anonymous) with value 66
- Node “x” with value 22

You should draw this as follows:

44(h) → 33(k) → 88 → 66 → 22(x)

21	`IntNode h = new IntNode(10,null);`
	10(h)
22	`h.addNodeAfter(30);`
	10(h) → 30
23	`h = new IntNode(20,h);`
	20(h) → 10 → 30
24	`h.addNodeAfter(40);`
	20(h) → 40 → 10 → 30
25	`IntNode n = IntNode.listSearch(h, 10);`
	20(h) → 40 → 10(n) → 30
26	`n.addNodeAfter(50);`
	20(h) → 40 → 10(n) → 50 → 30
27	`h.addNodeAfter(60);`
	20(h) → 60 → 40 → 10(n) → 50 → 30
28	`IntNode m= IntNode.listPosition(h, 4);`
	20(h) → 60 → 40 → 10(m) → 50 → 30
29	`m.addNodeAfter(70);`
	20(h) → 60 → 40 → 10 → 70 → 50 → 30
30	`m = h.getLink();`
	20(h) → 60(m) → 40 → 10 → 70 → 50 → 30
31	`m.removeNodeAfter();`
	20(h) → 60(m) → 10 → 70 → 50 → 30

Exercise 3: Implement `listRange` method

[4 pts] Implement a **`static`** method, called **`listRange`**, that takes one input parameter called head of type **`IntNode`** that represents the head of a linked list of ints. The method then finds and returns the difference between the maximum and minimum values in the linked list. For example, the range of the following linked list is 60 (which is 70 - 10). Assume the list include only positive integers. The methods returns -1 if the list is empty.

head → 20 → 60 → 10 → 50 → 70 → 30 → null

```java
public static int listRange(IntNode head) {
    int max = head.getData();
    int min = head.getData();
    int range; ;

    if (head.getLink() == null) {
        return -1;
    }

    else {
        for (IntNode cursor = head; cursor != null; cursor = cursor.getLink()) {
            if (cursor != null) {
                if (cursor.getData() > max) {
                    max = cursor.getData();
                }
                else if (cursor.getData() < min) {
                    min = cursor.getData();
                }
            }
        }
        range = max - min;
        return range;
    }
}
```

```java
Run | Debug
public static void main(String[] args) {
    IntNode head_null = new IntNode(initialData:0, initialLink:null);
    IntNode head = new IntNode(initialData:20, initialLink:null);
    head.addNodeAfter(element:30);
    head.addNodeAfter(element:70);
    head.addNodeAfter(element:50);
    head.addNodeAfter(element:10);
    head.addNodeAfter(element:60);

    for (IntNode cursor = head; cursor != null; cursor = cursor.getLink()) {
        System.out.print(cursor.getData() + " -> ");
    }

    System.out.println(x:"\n\nTesting non-null Linked List");
    System.out.println(listRange(head) + "\n");
    System.out.println(x:"Testing null Linked List");
    System.out.println(listRange(head_null));
}
```

```
20 -> 60 -> 10 -> 50 -> 70 -> 30 ->

Testing non-null Linked List
60

Testing null Linked List
-1
tony@Anthonys-MacBook-Pro week06_code %
```

Exercise 4: reflect on your learning this week

1- What is the most important thing you learned in this week's lecture? Write 2-3 sentences to explain your answer.

I did not realize that Linked Lists are a class, I now see how data Structures can be created. I was a bit confused before.

2- What is the muddiest (most-unclear) point(s) in this week's lecture? Explain why?

After the worksheet, this is pretty Clear. Understanding how cursor.getLinked() moved the Cursor forward was confusing at first.