**Christian Biermann**

**Problems**

The problem for this lab is to read a grocery list a perform specific task such as outputting the file, removing and accessing items within the list, check the total value of the list, and check if a certain item is within the list.

**Solution**

The solution to this problem is using a linked list and assigning each item and its value in the grocery list to a node. This would allow the data to be easily accessed and managed. Using File I/O will allow the program to read the file and output it. Along with this to remove and access items within the list will require a few methods such as: goToNext() which moves the current node to the next one after it, getCurrent() to access the current node, and removeCurrent() which will remove the current node. Checking the total value of the list is within the getTotal() method, where the program loops through the linked list and adds all of the values together. The same process exists when checking if an item is within the list, just when looping through the linked list, you compare the item you are looking for with each item in the list with the .equals() method.

**Problems Encountered**

There was only one problem that occurred when programming this, and that was the total output being wrong. This was easily fixed by changing the code within the while loop in the getTotal() method to add an if statement.

4. One advantage of a Linked List vs an Array, is that it is easier to add and remove elements to a linked list. You can use the goToNext() method or create a goTo() method which takes you to the item you want and can remove or add data after it with the addItem() and removeCurrent() methods. Another benefit of Linked List is the sizing of the list vs the array. With an array, you must predetermine the size of the array, but with Linked Lists, you do not have to initialize it with a static size. A third benefit is memory usage. Linked Lists are more efficient in using memory even though it uses more overall. The reason why is because Linked Lists uses memory for the nodes that are in use.

5. A disadvantage of a Linked Lists is it does not have random access like an Array. To access an element within the array, you must start at the head and move through the list. While it is an advantage as said, the list using more memory is still a disadvantage since it slows down the speed of the computer.

6. The reason it is not printing anything is because the Linked List is not being incremented to move to the next node. The way to fix this is to put temp = temp.link in the while loop but still outside the if statement.

7. The values are being removed because when saying head = newNode, it recreates the list with only the head as data and everything else is null. A way to fix this is assigning the newNode’s link to the current head, and then set the head of the list to the new newNode. So, before head = newNode, do newNode.link = head.

8. The reason why it is not removing anything is because there is a line missing in the code. Adding head = temp after temp = temp.link will remove the first element from the list.

9. The reason why the NullPointerException exists is because the while loop condition makes temp stop when it is null, so temp.link has no link that can be null. A fix to this when removing the last element in the list is to change the while condition to be temp.link.link. Temp.link.link grabs the second to last element in the list, and temp.link = null, removes the last element in the list.

10. There might be two errors in this, when testing this, it would not do the multiplication and addition, so I am going to list both errors I had. The first error which is the method returning 24 instead of 120, is because the while loop’s condition should not be temp.link and only be temp. If there is an addition and multiplication error, a simple fix is instead of saying ret \*= temp.data, you could use the code, ret += ret \* temp.data. This gives you 120.