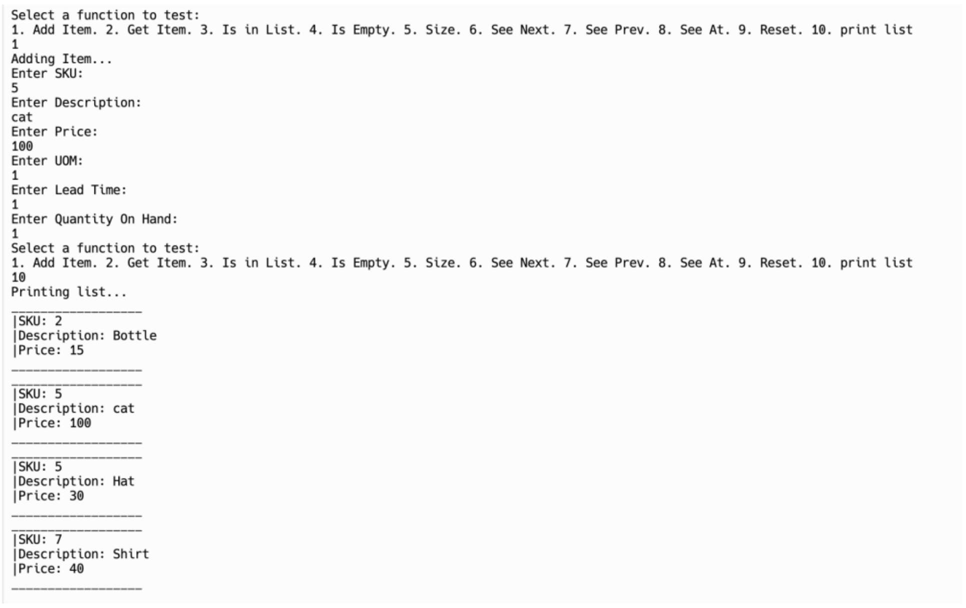
Lab 9 – Linked Lists

Maggie Lyon, Nyla Spencer, Ember Roberts

The concepts explored in this assignment include pointers, ordered linked lists, and overriding functions. These concepts are important when studying computer science/engineering since they are core concepts. Pointers allow for dynamic memory allocation, efficient memory management, and are fundamental in the implementation of data structures where nodes often reference other nodes. Ordered linked lists allow for searching and sorting in a memory efficient manner. They help organize data in a way that is efficient in accessing and manipulation.

These concepts were implemented in this lab by teh creating of an item class as well as a linked list class. Them linked list class stores items in order of the item member SKU. The main function allows the user to access the member functions of the ordered linked lists, as well as print ASCII art of it. The functionality of this class is shown in the screenshots below.

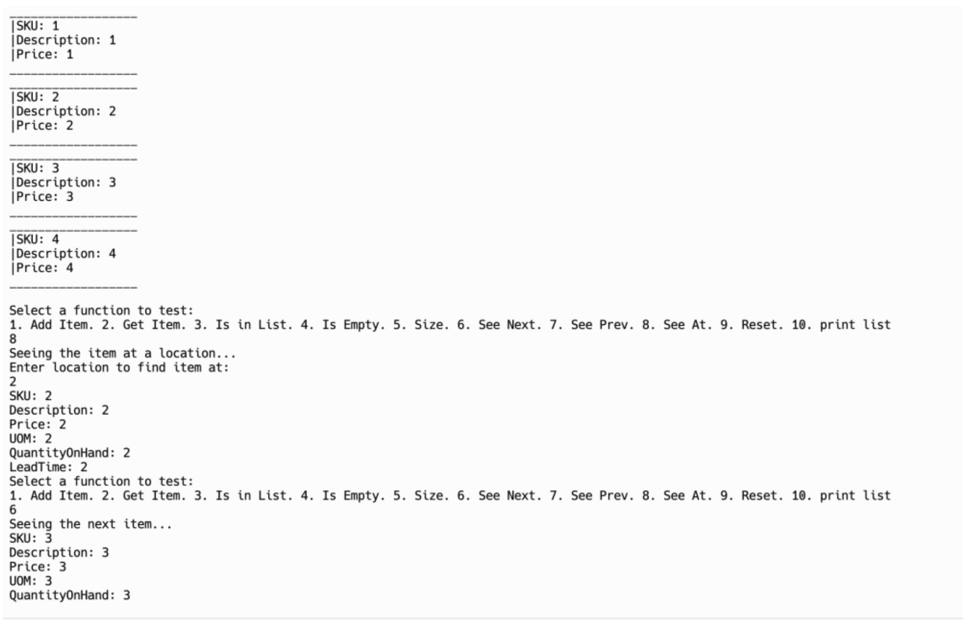
Add item/print list functionality



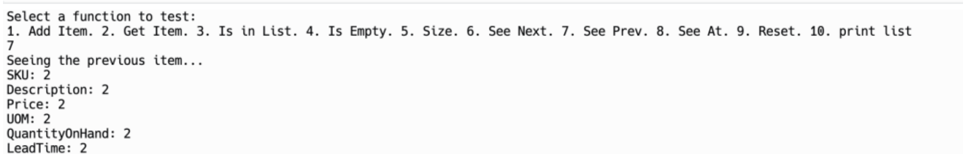
```
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
1
Adding Item...
Enter SKU:
5
Enter Description:
cat
Enter Price:
100
Enter UOM:
1
Enter Lead Time:
1
Enter Quantity On Hand:
1
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
10
Printing list...
__________________
|SKU: 2
|Description: Bottle
|Price: 15
__________________
__________________
|SKU: 5
|Description: cat
|Price: 100
__________________
__________________
|SKU: 5
|Description: Hat
|Price: 30
__________________
__________________
|SKU: 7
|Description: Shirt
|Price: 40
__________________
```

The screenshot above demonstrates the add item functionality as well as the print list functionality added in task 4. Four items were added to the list, and they were placed in order of SKU. When the print list function is called, they are printed in order in ASCII art.

See at/See Next/See Previous



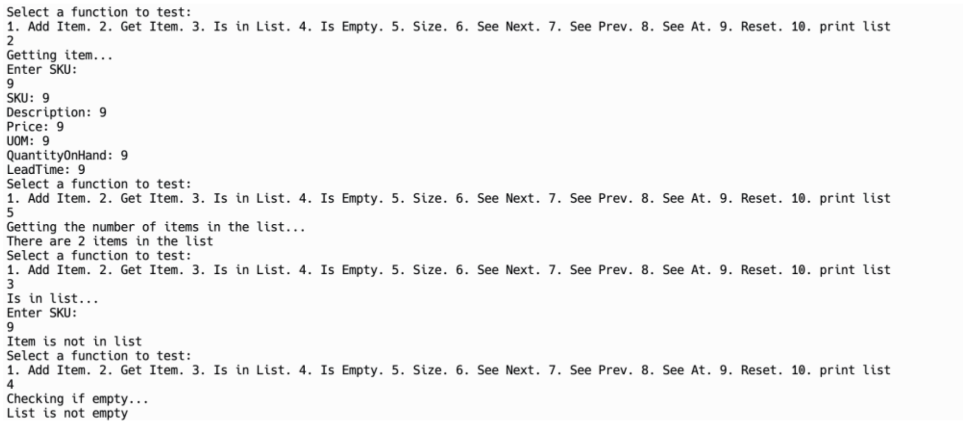
```
_________________
|SKU: 1
|Description: 1
|Price: 1
_________________
_________________
|SKU: 2
|Description: 2
|Price: 2
_________________
_________________
|SKU: 3
|Description: 3
|Price: 3
_________________
_________________
|SKU: 4
|Description: 4
|Price: 4
_________________

Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
8
Seeing the item at a location...
Enter location to find item at:
2
SKU: 2
Description: 2
Price: 2
UOM: 2
QuantityOnHand: 2
LeadTime: 2
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
6
Seeing the next item...
SKU: 3
Description: 3
Price: 3
UOM: 3
QuantityOnHand: 3
```



```
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
7
Seeing the previous item...
SKU: 2
Description: 2
Price: 2
UOM: 2
QuantityOnHand: 2
LeadTime: 2
```

In this screenshot, the database currently contains 4 items. We call the see at function to see the item in the second slot, and it returns the second item. Then, we call see next, and it returns the item at location 3. Finally, we call see previous to see the item at location 2.

Get item/ Size/ Is in List/ Is Empty



```
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
2
Getting item...
Enter SKU:
9
SKU: 9
Description: 9
Price: 9
UOM: 9
QuantityOnHand: 9
LeadTime: 9
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
5
Getting the number of items in the list...
There are 2 items in the list
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
3
Is in list...
Enter SKU:
9
Item is not in list
Select a function to test:
1. Add Item. 2. Get Item. 3. Is in List. 4. Is Empty. 5. Size. 6. See Next. 7. See Prev. 8. See At. 9. Reset. 10. print list
4
Checking if empty...
List is not empty
```

In this screenshot, an item with SKU 9 was perviously added to a list, and then removed with the get item function. The size function is called, and we see that htere are only two items in the list, and then we use the is in list function to check if 9 is one of them (it is not). Finally, we check if the list is empty (it is not).

Contributions:

Maggie – programming for main and item class

Ember – programming for linked list class