CS2028C Lab 8

Nicholas McClorey Daniel Wood, Kyle Van Blaricom February 25, 2019

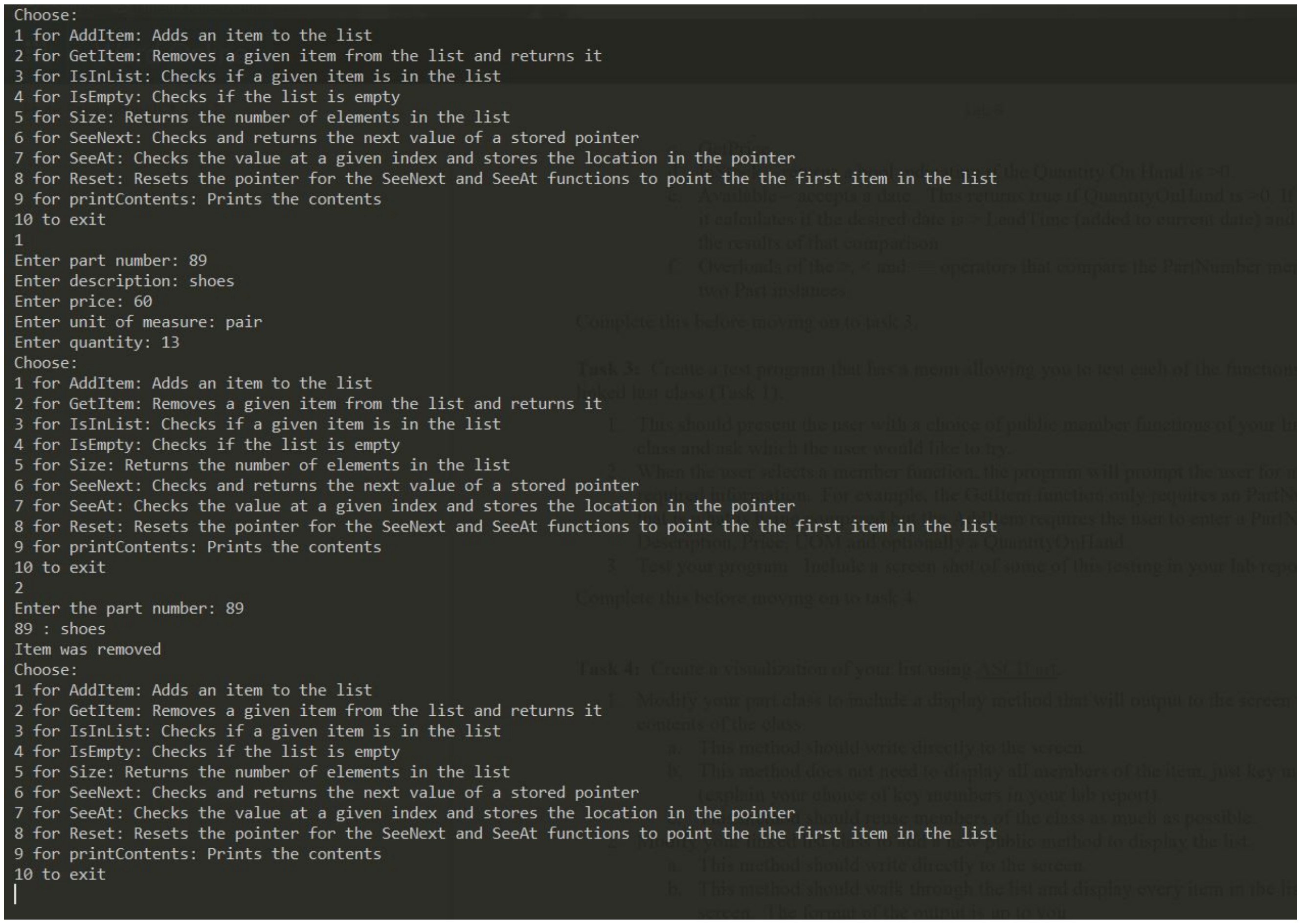
Overview

The purpose of the this lab was to create a linked list. Linked lists demonstrate how pointers can be used to create a list that performs more efficiently than other types of lists. Since all of the data isn't stored in sequential memory, items can be added, removed and inserted without shifting other items or reallocating memory for the entire list. This is ideal for lists that change size unpredictably. It also shows the ability of using data types that contain a pointer to another instance of the class. In our case, the "Node" class contained a pointer to another Node class. This could allow us to create all sorts of data structures that are connected by pointers.

Compiling

This program was compiled using gcc version 6.3.0. The tdm compiler can also be used to compile this program. The command to compile this program is "g++ -std=c++11 main.cpp Part.cpp". This program was compiled using Windows 10.

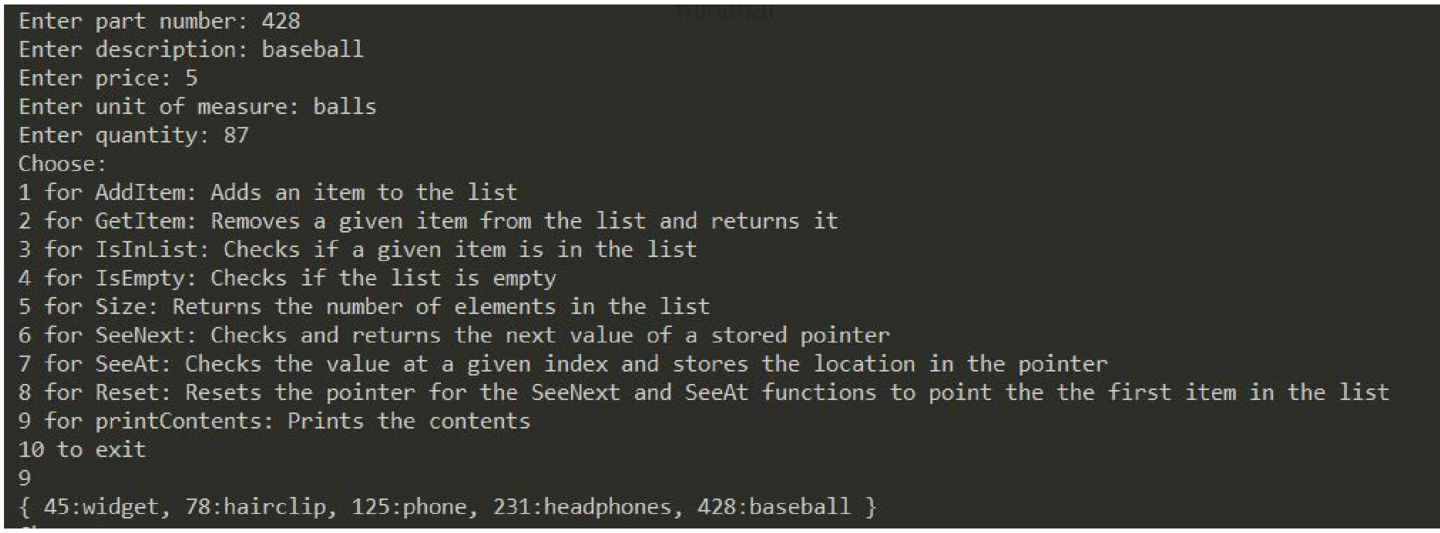
Task 3



```
Choose:
1 for AddItem: Adds an item to the list
2 for GetItem: Removes a given item from the list and returns it
3 for IsInList: Checks if a given item is in the list
4 for IsEmpty: Checks if the list is empty
5 for Size: Returns the number of elements in the list
6 for SeeNext: Checks and returns the next value of a stored pointer
7 for SeeAt: Checks the value at a given index and stores the location in the pointer
8 for Reset: Resets the pointer for the SeeNext and SeeAt functions to point the the first item in the list
9 for printContents: Prints the contents
10 to exit
1
Enter part number: 89
Enter description: shoes
Enter price: 60
Enter unit of measure: pair
Enter quantity: 13
Choose:
1 for AddItem: Adds an item to the list
2 for GetItem: Removes a given item from the list and returns it
3 for IsInList: Checks if a given item is in the list
4 for IsEmpty: Checks if the list is empty
5 for Size: Returns the number of elements in the list
6 for SeeNext: Checks and returns the next value of a stored pointer
7 for SeeAt: Checks the value at a given index and stores the location in the pointer
8 for Reset: Resets the pointer for the SeeNext and SeeAt functions to point the the first item in the list
9 for printContents: Prints the contents
10 to exit
2
Enter the part number: 89
89 : shoes
Item was removed
Choose:
1 for AddItem: Adds an item to the list
2 for GetItem: Removes a given item from the list and returns it
3 for IsInList: Checks if a given item is in the list
4 for IsEmpty: Checks if the list is empty
5 for Size: Returns the number of elements in the list
6 for SeeNext: Checks and returns the next value of a stored pointer
7 for SeeAt: Checks the value at a given index and stores the location in the pointer
8 for Reset: Resets the pointer for the SeeNext and SeeAt functions to point the the first item in the list
9 for printContents: Prints the contents
10 to exit
```

Task 4

When printing to the screen we chose to only print the part number and the description. We did this because these two properties exist to identify the part. The other properties are details that don’t say anything about the item itself. We also did this because it was concise and minimal.



```
Enter part number: 428
Enter description: baseball
Enter price: 5
Enter unit of measure: balls
Enter quantity: 87
Choose:
1 for AddItem: Adds an item to the list
2 for GetItem: Removes a given item from the list and returns it
3 for IsInList: Checks if a given item is in the list
4 for IsEmpty: Checks if the list is empty
5 for Size: Returns the number of elements in the list
6 for SeeNext: Checks and returns the next value of a stored pointer
7 for SeeAt: Checks the value at a given index and stores the location in the pointer
8 for Reset: Resets the pointer for the SeeNext and SeeAt functions to point the the first item in the list
9 for printContents: Prints the contents
10 to exit
9
{ 45:widget, 78:hairclip, 125:phone, 231:headphones, 428:baseball }
```