Amy Jiang
5/24/2017

Chapter 7 Exercises on page 251

#3b. Write the code to add a displayMethod to ADT stack (so it will be a member function) using the ***Linked Based implementation***. So add your displayMethod to the LinkedStack. Remember it must be a template. (10 points)

```
template<class ItemType>
void LinkedStack<ItemType>::display() const{
	Node<ItemType>* pointer = topPtr;
	cout << " | ";
	while (pointer != nullptr) {
		cout << pointer->getItem() << " | ";
		pointer = pointer->getNext();
	}
	cout << "\n";
}
```

#3c. Write the code to add a displayMethod to ADT stack (so it will be a member function) using the ***Array Based Implementation***. So add your displayMethod to the ArrayStack. Remember it must be a template. (10 points)

```
template<class ItemType>
void ArrayStack<ItemType>::display() const {
	cout << " | ";
	for (int i = 0; i <= top; i++) {
		cout << items[i] << " | ";
	}
	cout << "\n";
}
```

Amy Jiang
5/24/2017

Chapter 8 Exercises on Page 263

#1. Consider an ADT list instance of integers. Write a pseudocode function (for the client - so not a member function but a function for main) that computes the sum of the integers in the list aList. Make sure your function isn't dependent on how the list is implemented. (10 points)

```
#include <iostream>
#include <string>
#include "LinkedList.h" // ADT list operations
using namespace std;

int main()
{
	LinkedList<int> llist;
	LinkedList<int>* listptr = &llist;
	llist.insert(1,4);
	llist.insert(1,2);
	llist.insert(1,5);
	llist.insert(1, 66);

	double total = 0;
	for (int i = 0; i < llist.getLength(); i++) {
		total += listptr->getEntry(i + 1);
	}
	cout << "total: " << total << endl;

	system("PAUSE");
	return 0;
}  // end main
```

Amy Jiang
5/24/2017

#6. Write the pseudocode for a a client level function (so a function for main not the an ADT member function) named getPosition that returns the position of a given entry in a given list.

```
#include <iostream>
#include <string>
#include "LinkedList.h" // ADT list operations
using namespace std;

int main()
{
	LinkedList<int> llist;
	LinkedList<int>* listptr = &llist;
	llist.insert(1,4);
	llist.insert(2,2);
	llist.insert(3,5);
	llist.insert(4, 66);

	int numToCheck = 0;
	int position = 1;
	cout << "please enter the number to check: " << endl;
	cin >> numToCheck;

	for (int j = 0; j < llist.getLength(); j++) {
		if (listptr->getEntry(j + 1) == numToCheck) {
			position = ++j;
			break;
		}
		else position = -1;
	}
	cout << "position for number " << numToCheck << " and is: " << position << endl;

	system("PAUSE");
	return 0;
}  // end main
```