CSCI 2720 Data Structures
Assignment 1 – Linked List
DUE: October 1st, 2018 11:59PM

Summary:

In this assignment you will create a sorted Singly-Linked List that performs basic list operations. Each operation should be implemented in an abstract manner, so the data type for every operation should still accomplish the corresponding output. You will NOT use templates in this assignment. In order to accomplish this abstract implementation, you will create a class called DataType. DataType will store the actual data type that the list operations will be performed on, and in this specific assignment the type will be integers. DataType abstracts the data for the list by providing functions, such as compareTo and getvalue, that the linked list class should utilize. Once all of the linked list operations are implemented, you will create a Main application that initializes a list from an input file, and allows a user to interactively call the different list operations on it. The files you will create to accomplish this assignment are DataType.h, DataType.cpp, SortedLinkedList.h, SortedLinkedList.cpp, ListNode.h and Main.cpp. Finally, be sure to properly document all code with comments, and add your name above functions that you implement if you are in a group.

DataType.h _class should be composed of an enumeration called Comparison and the following function declarations:
Comparison with values GREATER, LESS, and EQUAL.

Public functions:

explicit DataType(int value)
Post-Condition: DataType object is created.

Comparison compareTo(DataType &item)
Pre-Condition: item parameter is initialized.
Post-Condition: returns an enumeration that indicates whether the calling object is GREATER, LESS, or EQUAL to item.

int getValue() const;
Pre-Condition: DataType object has been initialized.
Post-Condition: return the value instance variable.

Private data member:

int value

DataType.cpp should provide implementations for the above enumeration and functions.

ListNode.h should be composed of struct called ListNode.
ListNode should contain the members:

DataType item
ListNode *next
explicit ListNode(DataType &item) : item(item) {}

SortedLinkedList.h should be composed of the following function declarations:

public functions:

SortedLinkedList()
Post-Condition: List is created.

~SortedLinkedList()
Pre-Condition: List is created.
Post-Condition: all nodes are freed.

int length() const
Pre-Condition: List is initialized.
Post-Condition: return length instance variable.

void insertItem(DataType &item)
Pre-Condition: List exists.
Post-Condition: item is inserted into the list in the sorted order.

void deleteItem(DataType &item)
Pre-Condition: List exists and item initialized.
Post-Condition: the node that contains item is removed from the list.

int search(DataType &item)
Pre-Condition: List exists.
Post-Condition: the position of the node that contains the item is returned.

void clear()
Pre-Condition: List exists.
Post-Condition: List contains no items. Length is equal to 0. All node pointer memory freed.

For the Bonus Question: You may include the following prototype function:

void pairwiseSwap()
Pre-Condition: List exists.
Post-Condition: items in the list are sorted before swapping and the adjacent pair nodes are swapped according to their index position

private data members

int count
ListNode *head

SortedLinkedList.cpp should implement every struct and function listed in SortedLinkedList.h.

Main.cpp: Please use the following convention to design your function operation menu in your Main.cpp:

INSERT = 'i',
DELETE = 'd',
SEARCH = 's',

```
PRINT_ALL = 'p',
LENGTH = 'l',
CLEAR_ALL = 'c',
PAIR_SWAP = 'b',
QUIT = 'q'
```

PRINT_ALL: Items are printed to standard-out based on the implementation in DataType.
QUIT: It quits the program.

Example Output:

```
./main input.txt
Commands:
    (i) - Insert value
    (d) - Delete value
    (s) - Search value
    (p) - Print list
    (l) - Print length
    (b) - Pairwise Swap
    (c) - Clear list
    (q) - Quit program

Enter a command: i
1 3 9 10 19 37 45 63 84 100
Enter number: 12
1 3 9 10 12 19 37 45 63 84 100
----------------------------------------
Enter a command: d
1 2 3 9 10 12 19 37 45 63 84 100
Enter value to delete: 12
1 2 3 9 10 19 37 45 63 84 100
----------------------------------------
Enter a command: s
Enter a value to search: 10
Index of the item is 3
----------------------------------------
Enter a command: n
Next value is 1
Enter a command: n
Next value is 2
----------------------------------------
Enter a command: p
1 2 3 9 10 19 37 45 63 84 100
```

```
---------------------------------------
Enter a command: l
List Length is 11
---------------------------------------
Enter a command: b
Before pairwise swap
1 2 3 9 10 19 37 45 63 84 100
After pairwise swap
2 1 9 3 19 10 45 37 84 63 100
---------------------------------------
Enter a command: c
List cleared
---------------------------------------
Enter a command: q
Quitting program..
```

Bonus Question:

In the bonus question, you are expected to pairwise swap the linked list. Refer to the prototype given under SortedLinkedList.h ;

If the linked list is 1 2 4 5 6 7,
Then the output would be 2 1 5 4 7 6.

And if the linked list is 1 2 3 4 5 6 7,
Then the output would be 2 1 4 3 6 5 7.

Grading Rubric:

Operation	**Grade**
insertItem(DataType &item)	20 pts
deleteItem(DataType &item)	20 pts
searchItem(DataType &item)	20 pts
int length()	10 pts
void clear()	10 pts
void printList(SortedLinkedList &)	5 pts
Comment Lines	5 pts
Indentation	5 pts
Memory leakage	5 pts
BONUS question	**20 pts**
TOTAL	120 pts

Compiling:

A Makefile should compile your code into program called “main”. It should also contain an option called “run” that allows your application to be run with any text input file similar to input.txt.

Your program should be run with the following command syntax:

./main <input file name>

Code that fails to compile will receive a grade of a zero.

Late Policy:

Except in the cases of serious illness or emergencies, projects must be submitted before the specified deadline in order to receive full credit. Projects submitted late will be subject to the following penalties:

- If submitted 0–24 hours after the deadline 20% will be deducted from the project score
- If submitted 24–48 hours after the deadline 40% points will be deducted from the project score.
- If submitted more than 48 hours after the deadline a score of 0 will be given for the project.

Note that students who are unable to submit a project due to a serious illness or other emergency should contact the instructor as soon as possible before a project’s deadline. Based upon the circumstances, the instructor will decide an appropriate course of action.

Submission:

Create a Readme file with the instructions on how to compile your program. Submit the following files within a zipped folder on eLC:

DataType.h
DataType.cpp
SortedLinkedList.h
SortedLinkedList.cpp
ListNode.h
Main.cpp
Makefile
Readme