SNU CSE: Data Structure Due Date : 11 pm, Apr 06 2019

Assignment 1

Lecturer: Prof. Satti

Number 1: Linked List implementation (**55P**)

Instructions: Your task is to implement an interactive program that uses the Linked List data structure. The program should include an interactive menu that asks which operation you want to execute. The menu should look as follows:

```
A       Add ContactEntry                // Adds the contact at the specific index
L       Add Last                        // Adds the contact at the end of the list
E       Check if Empty
S       Search for Contact
P       Print Contacts
Z       Print Names
N       List Current Size
Q       Quit
R       Remove ContactEntry
T       Sort
U       Remove Duplicates
?       Display Help
```

Note:

a) The name of the main class is myAssignment

b) Create a myLinkedList class that implements LinearList interface (iterator must be used).

c) Your linked list should store objects of ContactEntry class. The fields are given (name, phoneNumber, occupation), but create your own constructor and getters/setters as needed.

d) You should handle both uppercase and lowercase character in the command prompt. That is, if somebody types 'a' you should consider it identical to what is done when typing 'A'.

e) Non existence character like "b" should be handled: ask `"What action would you like to perform?"` again .

f) When an action is finished, your program should reprompt `"What action would you like to perform?"` until the program is explicitly finished with 'Q', which should return `"Program Finished"`

Add Contact Entry(5p): the program asks users to input the name, phone number, and occupation. Then it creates an ContactEntry object and adds it to the arbitrary index of the list starting from 0 index.

- Note: You can not add to an index which the previous node does not exist, it means if you want to add an ContractEntry with index 10, the linked list should have at least index

9. You should handle the exception and print out "`The last index is __`" and ask the question again - "`Please enter an index to add.`"

- Format and example:

```
...
What action would you like to perform?
a
Please enter an index:
2
The last index is 0
Please enter an index:
1
Please enter a name:
Tom
Please enter a phone number:
01033333333
Please enter an occupation:
TA
What action would you like to perform?
...
```

Add Last(5p): add the arbitrary ContactEntry at the end of the list. You must implement a recursive solution.

- Note: If the list is empty, it should become the first element on the list.
- Format and example:

```
...
What action would you like to perform?
l
Please enter a name:
Tom
Please enter a phone number:
01033333333
Please enter an occupation:
TA
What action would you like to perform?
...
```

Check if Empty(5p): use isEmpty() function and if the list is empty print out "`The list is empty`" and if the list contains at least one element print out "`The list contains some element(s)`".

Search for Contact(5p): the program asks the index and should return the ContactEntry's name value at that index. You should take the index you want to search and the result should be something like: (`contact at the given index is Tom`) (6p)

- Note: if the index does not exist in the list return `the "index you want is not in the list"` and ask "`Please enter an index to search: `" again.

Print Contacts(5p): Print out in this format :

```
Students:
Name1 phone1
Name2 phone2
…

Professors:
Name3 phone3
Name3 phone3
…

TA:
```

- Note: an empty line is printed out below TA because there are no TA contacts on the list.

Print Names(5p): Print out the names of the ContactEntry objects in the list in its order (from index 0 to list size - 1).

- Note: Single space before the first element, in between, and at the end.
  Example: `{ John Mike Tom }`

List Current Size(5p): print out the current size of your list, Example: `The current size is 3`

Quit(2p): Terminate the program.

Remove Contact(5p): you should do everything needed for removing a ContactEntry of a given index on the list and print out "`'name' was removed`".

- Note: You should check whether the element in the index exists or not. If the index does not exist notify by "`It does not exist`" and ask the "`Please enter the index of the list to remove" again.` If the list is empty return "`Empty list`"

Sort(6p): You need to sort the list based on the ContactEntry's name in alphabetical order and print "`list sorted`". You do not need to print the content of the reversed list.

- Example : Tom , Mike , Allan - > Allan, Mike, Tom

Remove Duplicate(5p): remove duplicate nodes, keeping the first occurrence of each node.(8p)

- Note: the name, phoneNumber, and occupation should match to be considered duplicate. Return "`Duplicate contact removed`" or "`there is no duplicate`" if no duplicates exist.

Display Help(2p): show the menu again

Number 2: Arraylist vs Linked List (**20P**)

Implement the LinearList interface using an array; call this class myArrayList. Then make a main class called Compare in which you add 100,000 random strings (of any size) to both lists. Finally, compare the performances of get(int index), remove(int index), and add(int index, Object obj) for indices 1, 49999, and 99999. (10 points for adding 100,000 random strings, 10 points for printing out the performance results correctly.

Note:

a) The main class has name Compare
b) Print out as the example below
c) Unit of the displayed result is nanosecond. (Just perform the performance test once for each operation.)

Output example: (/* \t is a tab character. */)

```
[ds00@ds ~]$ java Compare
\tArrayList\tLinkedList\tIndex = 0
-----------------------------------------------
Add\t|11609741\t|1570250

Get\t|366421333\t|1456222

Remove\t|362122817\t|11609741


\tArrayList\tLinkedList\tIndex = 49,999
-----------------------------------------------
Add\t|11609741\t|1570250
```

```
Get\t|366421333\t|1456222

Remove\t|362122817\t|11609741


\tArrayList\tLinkedList\tIndex = 99,999
-----------------------------------------------
Add\t|11609741\t|1570250

Get\t|366421333\t|1456222

Remove\t|362122817\t|11609741
```

Number 3-1: Performance Explanation **(15P)**

Explain your results in Number 2 in a paragraph or two that answers the below questions. Also make a table of the worst time complexity for add(), remove(), and get(), for array linear list and (singly) linked linear list (see table below). You should submit for this problem a pdf file called Explanation.

- Did performance for each operation (add(), remove(), and get()) differ depending on the index? Why is that? (5pts)
- How would a doubly-linked list implementation of myLinkedList change the performance? (5pts)
-

	add()	remove()	get()
Array Linear List			
Linked Linear List			

(5pts)

Number 3-2 Arrange the following functions in the non-decreasing order of their asymptotic growth**: (10p)**

$$2^{3\log n},\ n^{\log n},\ n^2(\log n)^6,\ (1.1)^n,\ (\log n)^{(1/2)\log n},\ n^{2.5},\ (\log n)^{n/3},\ 4^{\sqrt{n}},\ 2^{2^{2\log\log n}},\ 6^{\log n}$$

Cooperation:

This assignment is to be done individually. Do not share code with others.

Assessment:

- Copying others work will result zero.
- If your program doesn't compile, then you will get a zero score.
- Grading will be done in the Linux environment with Java 11. If you comment or use any Korean character, there will be an error during process of compiling.

Submission:

- By eTL. Make sure you write your name and student number as a comment on top of myAssignment.java and Compare.java.
- Compress your ContactEntry.java, LinearList.java, myLinkedList.java, myArrayList.java, Compare.java, myAssignment.java, and Explanation.pdf as a zip file.
- Do NOT email for changing in the assignment after you have submitted.
- Late submissions will not be accepted