DEPARTMENT OF COMPUTER SCIENCE
CPSC 101 – Computer Programming II

Assignment 4

Due Date: April 19, 2021 at 11:59 PM

Notes:

- Zip ALL your `.java` files together into one package named: "`CPSC101_A#_lastname_firstname_student#.zip`" (or .rar, or .7z).
- Submit your assignment on Blackboard course homepage.

A Floating Point Linked List

Background

We implemented an integer linked list together in class. In this assignment, your task is to implement a floating point linked list.

A driver program is provided. Your implementation of the linked list must work with the driver program without errors, and provide the same output as shown in the sample output section. And of course, you should **NOT** make any modifications in the driver program.

Instructions

The driver program contains many tests of different features/methods of a linked list, which includes:

- Utility Methods
  1. `isEmpty()`: Returns `true` if the list is currently empty, `false` otherwise.
  2. `getSize()`: Returns the size of the linked list.
  3. `toString()`: Returns a string representation of the linked list.
- Insertion Methods
  4. `append(MyNode n)`: Appends a new node at the end of the list.
  5. `prepend(MyNode n)`: Prepends a new node at the beginning of the list.
  6. `insert(MyNode n, int index)`: Insert a new node at a specific index.
- Deletion Methods
  7. `remove(int index)`: Removes a node from a specific index.
  8. `delete(MyNode n)`: Deletes a specific node from the list.
     a. Do nothing if the node does not exist.
  9. `reset()`: Empties the list.
- Search Methods
  10. `search(double key)`: Searches list with a specific key. Returns a reference to the node if it is found, otherwise, returns `null`.

11. `getMyNodeAt(int index)`: Searches list with a specific index. Returns a reference to the node at the index. If the index cannot be reach, return `null`.
12. `getIndexOf(double key)`: Searches list for a specific key. Returns the index of the first appearance of the node that contains the key. If the key cannot be found, returns `-1`.

- Other Methods
  13. `sort()`: Sort the list in ascending order.
  14. Feel free to implement other methods if it is necessary, e.g., swap nodes, etc.

Sample Output

```
================================
Testing Program for MyLinkedList
================================

================================
Test 1: Testing Insertion on Empty List...
Node [ 1.0 ] ->  does not exist!
Cannot remove node at [0]! List is empty!
================================
Generating Original List...
Done!
The Original List is:
[ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] ->
================================
Test 2: Testing Insertion...
Insert Node 1.0
[ 1.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] ->
Insert Node 0.0 at index 0
[ 0.0 ] -> [ 1.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] ->
Insert Node 2.0 at index 2
[ 0.0 ] -> [ 1.0 ] -> [ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] ->
Insert Node 10.0 at index (size of the current list)
[ 0.0 ] -> [ 1.0 ] -> [ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] -> [ 10.0 ] ->
================================
Test 3: Testing Deletion...
Delete Node 1.0
[ 0.0 ] -> [ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] -> [ 10.0 ] ->
Delete Node 99.0
Node [ 99.0 ] ->  does not exist!
[ 0.0 ] -> [ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] -> [ 10.0 ] ->
Remove Node at index 0
[ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] -> [ 10.0 ] ->
Remove Node at index 99
Cannot remove node at [99]!
[ 2.0 ] -> [ 7.0 ] -> [ 6.0 ] -> [ 5.0 ] -> [ 4.0 ] -> [ 3.0 ] -> [ 10.0 ] ->
================================
Test 3: Testing Search...
Search Node with data 4.0
[ 4.0 ] ->
Search Node with data 99.0
null
Search Node at index 0
[ 2.0 ] ->
Search Node at index (size of the current list - 1)
[ 10.0 ] ->
Search Node at index 99
null
Get the index of Node 2.0
0
Get the index of Node 99.0
-1
================================
Test 4: Testing Sort...
Sorting the list...
[ 2.0 ] -> [ 3.0 ] -> [ 4.0 ] -> [ 5.0 ] -> [ 6.0 ] -> [ 7.0 ] -> [ 10.0 ] ->

End of Testing...
```