**COMP 2150 – Fall 2019**

**Homework 7: LinkedList,DoublyLinkedlist,**

**( 67 points)**

Number of People: Individual. Feel free to ask me for help, or visit the Computer Science Learning Center (<http://www.memphis.edu/cs/current_students/cslc.php>).

Due: Sun., Nov 17, 2019 by 11:59 pm

Submission: Zip all your Java source files (you can zip the entire project folder if using an IDE) into a single file and upload it to the proper folder in the eLearn dropbox at <https://elearn.memphis.edu>. **Do not include Password protection on you zipped file. You have to include this question paper with answers to the submitted ZIP file**

Coding Style: Use consistent indentation. Use standard Java naming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT\_NAMES**. Include a reasonable amount of comments.

Grader:

Naeem Khoshnevis ([nkhshnvs@memphis.edu](mailto:nkhshnvs@memphis.edu)) (Section 001)

Utsav Shrestha ([shrstha3@memphis.edu](mailto:shrstha3@memphis.edu) ) (Section 002)

Questions about grading? Please contact Naeem (001) or Utsav(002)

This assignment deals with writing programs in the areas of linkedList (working with Nodes) and performance in DoublyLinkedList.

**// The following ListNode class is used for Question1, and 2)**

**//ListNode is a class for storing a single node of a linked list storing**

**//integer values. It has two public data fields for the data and the link to**

**//the next node in the list and has three constructors:**

**public** **class** ListNode **{**

**public** **int** data**;**

**public** ListNode next**;**

**// post: constructs a node with data 0 and null link**

**public** ListNode**()** **{**

**this(**0**,** **null);**

**}**

**// post: constructs a node with given data and null link**

**public** ListNode**(int** data**)** **{**

**this(**data**,** **null);**

**}**

**// post: constructs a node with given data and given link**

**public** ListNode**(int** data**,** ListNode next**)** **{**

**this.**data **=** data**;**

**this.**next **=** next**;**

**}**

**}**

***(For questions 1, you will have to Cut&Paste the “text-graphics” from the question to generate answers and change the node contents number (integer))***

1. (9 points) What would the given linked node diagram look like after the given code executes:
   1. (3 points)

**+----+----+ +----+----+**

**list ----> | 1 | +----> | 2 | / |**

**+----+----+ +----+----+**

List.next = new LinkNode(9);

*+----+----+ +----+----+*

*list ----> | 1 | +----> | 9 | / |*

*+----+----+ +----+----+*

* 1. (3 points)

**+----+----+ +----+----+**

**list ----> | 1 | +----> | 2 | / |**

**+----+----+ +----+----+**

List.next = new LinkNode(9, list.next);

*+----+----+ +----+----+ +----+----+*

*list ----> | 1 | +----> | 9 | +----> | 2 | / |*

*+----+----+ +----+----+ +----+----+*

* 1. (3 points)

+----+----+ +----+----+ +----+----+

list ----> | 7 | +----> | 8 | +----> | 3 | / |

+----+----+ +----+----+ +----+----+

List = new LinkNode(5, list.next.next);

*+----+----+ +----+----+*

*list ----> | 5 | +----> | 3 | / |*

*+----+----+ +----+----+*

1. (22 points) Write the code that will convert the given **Before links** to the **After links** with comments.

**For example:**

**Before:**

+----+----+ +----+----+

list ----> | 1 | +----> | 2 | / |

+----+----+ +----+----+

**After**:

+----+----+ +----+----+ +----+----+

list ----> | 1 | +----> | 2 | +----> | 3 | / |

+----+----+ +----+----+ +----+----+

**Code: (example, you must include comments to your code, otherwise your will only get 0.5 for the total marks)**

**list.next.next = new ListNode(3, null); // 2 -> 3**

1. **(3 points)**

**Before:**

+----+----+ +----+----+

list ----> | 1 | +----> | 2 | / |

+----+----+ +----+----+

**After:**

+----+----+ +----+----+ +----+----+

list ----> | 3 | +----> | 1 | +----> | 2 | / |

+----+----+ +----+----+ +----+----+

**Code: ( 3 points)**

*list = new ListNode(3, list) // (1 -> 2) -> (3 -> 1 -> 2)*

1. **(3 points)**

**Before:**

+----+----+ +----+----+

list ----> | 1 | +----> | 2 | / |

+----+----+ +----+----+

+----+----+ +----+----+

temp ----> | 3 | +----> | 4 | / |

+----+----+ +----+----+

**After:**

+----+----+ +----+----+ +----+----+ +----+----+

list ----> | 1 | +----> | 3 | +----> | 4 | +- --> | 2 | /

+----+----+ +----+----+ +----+----+ +----+----+

Code: ( 3 points)

*temp.next.next = list.next // (3 -> 4) -> (3 -> 4 -> 2)*

*list.next = temp // (1 -> 2) -> (1 -> 3 -> 4 -> 2)*

1. **(5 Points)**

Before

+----+----+ +----+----+ +----+----+

list ----> | 1 | +----> | 2 | +----> | 3 | / |

+----+----+ +----+----+ +----+----+

After:

**+----+----+**

**list ----> | 2 | / |**

**+----+----+**

**+----+----+ +----+----+**

**List2 ---> | 1 | +----> | 3 | / |**

**+----+----+ +----+----+**

Code: ( 5 points)

*ListNode list2 = new ListNode(list.data, list.next.next); // () -> (1 -> 3)*

*list = list.next; // (1 -> 2 -> 3) -> (2 -> 3)*

*list.next = null; // (2 -> 3) -> (2)*

1. **(5 points)**

**Before**

**+----+----+ +----+----+ +----+----+**

**list ----> | 5 | +----> | 4 | +----> | 3 | / |**

**+----+----+ +----+----+ +----+----+**

**After:**

**+----+----+ +----+----+ +----+----+**

**list ----> | 3 | +----> | 4 | +----> | 5 | / |**

**+----+----+ +----+----+ +----+----+**

Code: ( 5 points)

***int temp1 = list.data; // Create a temporary var to store ‘3’***

***int temp2 = list.next.next.data; // Create a temporary var to store ‘5’***

***list.next.next.data = temp1; // (3 -> 4 -> 5) -> (3 -> 4 -> 3)***

***list.data = temp2; // (3 -> 4 -> 3) -> (3 -> 4 -> 5)***

1. **(6 points)**

**Before:**

**+----+----+**

**list ----> | 1 | / |**

**+----+----+**

**+----+----+ +----+----+ +----+----+**

**list2 ---> | 2 | +----> | 3 | +----> | 4 | / |**

**+----+----+ +----+----+ +----+----+**

**After:**

**+----+----+ +----+----+ +----+----+**

**List ---> | 4 | +----> | 1 | +----> | 2 | / |**

**+----+----+ +----+----+ +----+----+**

**+----+----+**

**List2 ---> | 3 | / |**

**+----+----+**

Code: (6 points)

*int temp1 = list.data; // Create a temporary var to store ‘1’*

*int temp2 = list2.next.data; // Create a temporary var to store ‘3’*

*list = list2.next.next; // (1) -> (4)*

*list.next = new ListNode(temp1); // (4) -> (4 -> 1)*

*list.next.next = new ListNode(list2.data); // (4 -> 1) -> (4 -> 1 -> 2)*

*list2 = new ListNode(temp2); // (2 -> 3 -> 4) -> (3)*

1. (36 points) For this question, you will convert the given MyDoublyLinkedList.java (written for singly linkedList) that we discussed in class, to support a doubly linked list operation. The given MyDoublyLinkedList.java file is written in a singly LinkedList approach. Most of code has already been written. You will need to add extra codes to the myDoublyLinkedList.java file to support the doubly linked list approach in traversing through the list. In order to show that the doubly linked list operation is faster that the (given) singly operation, add some time measurement codes to compare the time performance of the two operations. At the time of writing this, the time measurement codes are included in the test code. (**There are 12 (3 points each) possible locations** in the program that that require you to make changes to support doubly linked list operation)

Before making any changes to your codes, you need to run the test program (TestDoublyLinkedList.java) and you should see the following:

*Notice the DURATION (msec) result on the 6th line from the end of the result display. After adding your doubly linked list codes, you should see improvement with the duration (by about 3x to 4x time faster). It depends on the kind of computer that you are using to run this program, the duration may be different. Everything else remains the same. You should run the test codes continuously after each modification to your codes. Be sure to have a backup of the original codes and test your codes every time after you have added a new doubly linked capability. For this problem, you do not have to change the TestDoublyLinkedList.java code. Unless you have discovered defects that require repair and please let me know.*

Initialized with {A,B,A,A}: [A, B, A, A]

Adding elements: [A, B, A, A]

- '\*' at first [\*, A, B, A, A]

- 'C' at last [\*, A, B, A, A, C]

- 'D' [\*, A, B, A, A, C, D]

- '#' @ 2 [\*, A, #, B, A, A, C, D]

Getting elements: [\*, A, #, B, A, A, C, D]

- First Element \*

- Last Element D

- Element at 1 A

- Element at 20 null

Setting elements: [\*, A, #, B, A, A, C, D]

- Element @ 0: '\*' -> '+' [+, A, #, B, A, A, C, D]

- Element @ 2: '#' -> '-' [+, A, -, B, A, A, C, D]

- Element @ 9: 'null' -> '&' [+, A, -, B, A, A, C, D]

Removing elements: [+, A, -, B, A, A, C, D]

- First element '+' [A, -, B, A, A, C, D]

- Last element 'D' [A, -, B, A, A, C]

- Element @ 1 '-' [A, B, A, A, C]

- Element @ 9 'null' [A, B, A, A, C]

Checking: [A, B, A, A, C]

- Contains 'A'? true

- Contains 'Z'? false

- First occurence of 'A' @ 0

- Last occurence of 'A' @ 3

- First index of 'Z' @ -1

- Last index of 'Z' @ -1

DURATION = 36.529286 msecs

ToString: [A, B, A, A, C]

ToReversedString: null

Adding 'X' at : 20

size of list = 5

[A, B, A, A, C, X]