CS 323\_33 Programming Language: Java

Project #2 Reversed Linked List

Andres Quintero

Due Date:

Soft copy: 02/10/2020

Hard copy: 02/13/2020

Submission Date:

Soft copy: 02/10/2020

Hard copy: 02/13/2020

Algorithm steps:

Main():

Step 0: Open input file and output files from command line arguments

Step 1: Initialize listHead as dummy node will values (“dummy”, NULL)

Step 2: Call constructLL() method

Step 3: Call printList() method

Step 4: Call reverseLL() method

Step 5: Call printList() method again

Step 6: Close all files

**Source code: reverseLL.java**

import java.util.\*;

import java.io.\*;

class listNode {

String data;

listNode next;

// Constructor to create a new node

// Next is by default initialized

// as null

listNode(String data, listNode next) {

this.data = data;

this.next = next;

}

}

public class reverseLL{

public static void main(String[] args) {

// Opening Files

Scanner inFile = null;

PrintWriter outFile1 = null;

PrintWriter outFile2 = null;

// Error Catching

try {

inFile = new Scanner(new File(args[0]));

} catch (FileNotFoundException err) {

System.out.println("Error in opening inputFile: " + err);

}

try {

outFile1 = new PrintWriter(args[1]);

} catch (FileNotFoundException err) {

System.out.println("Error in opening outFile1: " + err);

}

try {

outFile2 = new PrintWriter(args[2]);

} catch (FileNotFoundException err) {

System.out.println("Error in opening outFile2: " + err);

}

// Creating listHead

listNode listHead = new listNode("dummy", null);

constructLL(listHead, inFile, outFile2);

//Caption 1

outFile1.println("Printing the original sorted list");

printList(listHead, outFile1);

LLreverse(listHead, outFile2);

//Caption 2

outFile1.println("Printing the reversed sorted list");

printList(listHead, outFile1);

// Closing Files

inFile.close();

outFile1.close();

outFile2.close();

}

// Functions

public static void constructLL(listNode listHead, Scanner inFile, PrintWriter outFile) {

String data;

while(inFile.hasNext()){

data = inFile.next();

listNode newNode = new listNode(data, null);

listInsert(listHead, newNode);

printList(listHead, outFile);

}

}

public static void listInsert(listNode listHead, listNode newNode) {

listNode spot = findSpot(listHead, newNode);

newNode.next = spot.next;

spot.next = newNode;

}

public static listNode findSpot(listNode listHead, listNode newNode) {

listNode spot = listHead;

//compareToIgnoreCase because 'A' should be equal to 'a'

while(spot.next != null && ( spot.next.data.compareToIgnoreCase(newNode.data) < 0 )){

spot = spot.next;

}

return spot;

}

// Doesn't need to be listHead, node is okay

public static void printList(listNode node, PrintWriter outFile) {

outFile.print("listHead-> ");

while(node != null){

if(node.next == null){ // Reading null.data causes an error

outFile.print("(" + node.data + "," + node + "," + node.next + "," + "NULL" + ")");

} else {

outFile.print("(" + node.data + "," + node + "," + node.next + "," + node.next.data + ")-> ");

}

node = node.next;

}

outFile.println("-> NULL");

}

public static void LLreverse(listNode listHead, PrintWriter outFile) {

listNode last = listHead.next;

listNode spot = null;

while(last.next != null){

spot = last.next;

last.next = spot.next; // Need this else it hangs bc of unlinking

moveSpotNodeToFront(listHead, spot);

printList(listHead, outFile);

}

}

public static void moveSpotNodeToFront(listNode listHead, listNode spot) {

spot.next = listHead.next;

listHead.next = spot;

}

public static listNode findMiddleNode(listNode listHead, PrintWriter outFile) {

listNode walk1 = listHead;

listNode walk2 = listHead;

while(walk2 != null && walk2.next != null) {

printNode(walk1, outFile);

walk1 = walk1.next;

walk2 = walk2.next.next;

}

return walk1;

}

public static void printNode(listNode node, PrintWriter outFile) { // Same as printList except no loop

if(node.next == null){ // Reading null.data causes an error

outFile.print("(" + node.data + "," + node + "," + node.next + "," + "NULL" + ")");

} else {

outFile.print("(" + node.data + "," + node + "," + node.next + "," + node.next.data + ")->");

}

}

}

**Input File: LLReverse\_Data.txt**

Hishaam Esteban Kevin

Matthew Brandon

Joel Luis Jianwei Yechiel

Taeyong

Jiayu

Jiade Phillip

Russell Mohebullah

Akshar Evgeniia Andres Marco Justin

Robin Kelvin Zhiheng Jeffrey

Yifei Yinyu

Jiaxin Youyia Eleftherios

Yuan

**Output File 2: Print Linked List and Revered Linked List**

Printing the original sorted list

listHead-> (dummy,listNode@404b9385,listNode@3feba861,Akshar)->(Akshar,listNode@3feba861,listNode@6f496d9f,Andres)-> (Andres,listNode@6f496d9f,listNode@378bf509,Brandon)-> (Brandon,listNode@378bf509,listNode@12edcd21,Eleftherios)-> (Eleftherios,listNode@12edcd21,listNode@4dd8dc3,Esteban)-> (Esteban,listNode@4dd8dc3,listNode@5b480cf9,Evgeniia)-> (Evgeniia,listNode@5b480cf9,listNode@6d311334,Hishaam)-> (Hishaam,listNode@6d311334,listNode@1d81eb93,Jeffrey)-> (Jeffrey,listNode@1d81eb93,listNode@15aeb7ab,Jiade)-> (Jiade,listNode@15aeb7ab,listNode@16b98e56,Jianwei)-> (Jianwei,listNode@16b98e56,listNode@7cc355be,Jiaxin)->(Jiaxin,listNode@7cc355be,listNode@4f3f5b24,Jiayu)-> (Jiayu,listNode@4f3f5b24,listNode@5fd0d5ae,Joel)-> (Joel,listNode@5fd0d5ae,listNode@10f87f48,Justin)-> (Justin,listNode@10f87f48,listNode@2f4d3709,Kelvin)-> (Kelvin,listNode@2f4d3709,listNode@6d03e736,Kevin)-> (Kevin,listNode@6d03e736,listNode@2d98a335,Luis)-> (Luis,listNode@2d98a335,listNode@723279cf,Marco)-> (Marco,listNode@723279cf,listNode@568db2f2,Matthew)-> (Matthew,listNode@568db2f2,listNode@5f184fc6,Mohebullah)-> (Mohebullah,listNode@5f184fc6,listNode@7b23ec81,Phillip)-> (Phillip,listNode@7b23ec81,listNode@b4c966a,Robin)-> (Robin,listNode@b4c966a,listNode@6acbcfc0,Russell)-> (Russell,listNode@6acbcfc0,listNode@27d6c5e0,Taeyong)-> (Taeyong,listNode@27d6c5e0,listNode@7ef20235,Yechiel)-> (Yechiel,listNode@7ef20235,listNode@7291c18f,Yifei)-> (Yifei,listNode@7291c18f,listNode@34a245ab,Yinyu)-> (Yinyu,listNode@34a245ab,listNode@6e8cf4c6,Youyia)-> (Youyia,listNode@6e8cf4c6,listNode@34c45dca,Yuan)-> (Yuan,listNode@34c45dca,listNode@4e50df2e,Zhiheng)-> (Zhiheng,listNode@4e50df2e,null,NULL)-> NULL

Printing the reversed sorted list

listHead-> (dummy,listNode@404b9385,listNode@4e50df2e,Zhiheng)-> (Zhiheng,listNode@4e50df2e,listNode@34c45dca,Yuan)-> (Yuan,listNode@34c45dca,listNode@6e8cf4c6,Youyia)-> (Youyia,listNode@6e8cf4c6,listNode@34a245ab,Yinyu)-> (Yinyu,listNode@34a245ab,listNode@7291c18f,Yifei)-> (Yifei,listNode@7291c18f,listNode@7ef20235,Yechiel)-> (Yechiel,listNode@7ef20235,listNode@27d6c5e0,Taeyong)-> (Taeyong,listNode@27d6c5e0,listNode@6acbcfc0,Russell)-> (Russell,listNode@6acbcfc0,listNode@b4c966a,Robin)-> (Robin,listNode@b4c966a,listNode@7b23ec81,Phillip)-> (Phillip,listNode@7b23ec81,listNode@5f184fc6,Mohebullah)-> (Mohebullah,listNode@5f184fc6,listNode@568db2f2,Matthew)-> (Matthew,listNode@568db2f2,listNode@723279cf,Marco)-> (Marco,listNode@723279cf,listNode@2d98a335,Luis)-> (Luis,listNode@2d98a335,listNode@6d03e736,Kevin)-> (Kevin,listNode@6d03e736,listNode@2f4d3709,Kelvin)-> (Kelvin,listNode@2f4d3709,listNode@10f87f48,Justin)-> (Justin,listNode@10f87f48,listNode@5fd0d5ae,Joel)-> (Joel,listNode@5fd0d5ae,listNode@4f3f5b24,Jiayu)-> (Jiayu,listNode@4f3f5b24,listNode@7cc355be,Jiaxin)-> (Jiaxin,listNode@7cc355be,listNode@16b98e56,Jianwei)-> (Jianwei,listNode@16b98e56,listNode@15aeb7ab,Jiade)-> (Jiade,listNode@15aeb7ab,listNode@1d81eb93,Jeffrey)-> (Jeffrey,listNode@1d81eb93,listNode@6d311334,Hishaam)-> (Hishaam,listNode@6d311334,listNode@5b480cf9,Evgeniia)-> (Evgeniia,listNode@5b480cf9,listNode@4dd8dc3,Esteban)-> (Esteban,listNode@4dd8dc3,listNode@12edcd21,Eleftherios)-> (Eleftherios,listNode@12edcd21,listNode@378bf509,Brandon)-> (Brandon,listNode@378bf509,listNode@6f496d9f,Andres)-> (Andres,listNode@6f496d9f,listNode@3feba861,Akshar)-> (Akshar,listNode@3feba861,null,NULL)-> NULL

**Output File 2: Debugging Print – *Not Submitted***