Cheewin Thawornjaroenpong

CSCI323.25 Designs and Analysis of Algorithms (Spring 2023)

Project1

Ordered linked list

02/08/2023

Algorithm Steps:
Step 1: inFile open with args[0] outFile open with args[1] deBugFile open with args[2]
Step 2: listHead get a new listNode with (“dummy”), as the dummy node for listHead to point to.
Step 3: constructLL (listHead, inFile, deBugFile)
Step 4: printList (listHead, outFile) // Print the complete list to outFile
Step 5: middleNode findMiddleNode (listHead, deBugFile)
Step 6: if middleNode != null // in case the list is empty outFile middleNode’s data // with caption “the word in the middle of list is”
Step 7: Close all files

Illustrations:

Source code:

```
import java.io.*;
import java.util.*;

class ThawornjaroenpongC_Project1_Main {
   public static void main(String args []){

      try{
         Scanner inFile = new Scanner(new FileReader(args[0]));
         BufferedWriter outFile = new BufferedWriter(new FileWriter(args[1]));
         BufferedWriter deBugFile = new BufferedWriter(new FileWriter(args[2]));

         listNode listHead = new listNode("dummy");
         LList list = new LList().constructLL(listHead, inFile, deBugFile);

         list.printList(listHead, outFile);

         if(list.findMiddleNode(listHead, deBugFile)!= null)
         {
                  listNode middleNode = list.findMiddleNode(listHead, deBugFile);
                  outFile.write("The word in the middle is: " + middleNode.data);
         }

         inFile.close();
         outFile.close();
         deBugFile.close();

      }catch (IOException e) {
       // TODO Auto-generated catch block
         e.printStackTrace();
      }
```

```
}

static class listNode {
 String data;
 listNode next;

      listNode(){}
   listNode(String d)
   {
      data = d;
      next = null;
   }
}

static class LList {



   public LList constructLL(listNode listHead, Scanner inFile, BufferedWriter deBugFile)
   {
     try{
        deBugFile.write("In constructLL method");
        deBugFile.write("\n");
        while (inFile.hasNext())
           {
              String data = inFile.next();
              listNode newNode = new listNode(data);
              listInsert(listHead, newNode, deBugFile);
              printList(listHead, deBugFile);
           }
     }catch (IOException e) {
     // TODO Auto-generated catch block
      e.printStackTrace();
    }
     return this;
   }

   public void listInsert(listNode listHead, listNode newNode, BufferedWriter deBugFile)
   {
      try {
                                    deBugFile.write("In listInsert method");
```

```
                                deBugFile.write("\n");
                                listNode spot = findSpot(listHead, newNode);
                                deBugFile.write("Returns from findSpot where Spot.data is " +
spot.data);
                                deBugFile.write("\n");
                                deBugFile.write("newNode.data is " + newNode.data);
                                deBugFile.write("\n");

                                newNode.next = spot.next;
                                spot.next = newNode;


                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
        }

        public listNode findSpot(listNode listHead, listNode newNode)
        {
           listNode spot = listHead;

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

                spot = spot.next;
           }

                        return spot;

        }

        public void printList(listNode listHead, BufferedWriter File)
        {
           int count = 0;
           listNode tmp = listHead;

           try {
                while(tmp.next != null)
                {
                                File.write("(" + tmp.data + ", " + tmp.next.data + ")"
                                                + " -> ");
                                tmp = tmp.next;
```

```
                        count++;

                        if(count >= 5)
            {
                    File.write("\n");
                    count = 0;
            }
            }

            File.write("NULL");
            File.write("\n");

    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
}

public listNode findMiddleNode(listNode listHead, BufferedWriter deBugFile)
{
    listNode walker1 = listHead.next;
                    listNode walker2 = listHead.next;
    try {
                        deBugFile.write("In findMiddleNode method");
                        deBugFile.write("\n");


                        while(walker2 != null && walker2.next != null)
                        {
                                walker1 = walker1.next;
                                walker2 = walker2.next.next;

                                deBugFile.write("walker1's data is " + walker1.data);
                                deBugFile.write("\n");
                        }

                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

    return walker1;
}
```

```
    }
}
```

Program output:

outFile from LLMiddleNode_Data1 :

(dummy, are) -> (are, all) -> (all, and) -> (and, a) -> (a, ago) ->
(ago, and) -> (and, brought) -> (brought, created) -> (created, conceived) -> (conceived, continent) ->
(continent, dedicated) -> (dedicated, equal) -> (equal, forth) -> (forth, fathers) -> (fathers, Four) ->
(Four, in) -> (in, liberty) -> (liberty, men) -> (men, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, proposition) -> (proposition, seven) -> (seven, score) ->
(score, that) -> (that, the) -> (the, to) -> (to, this) -> (this, years) ->
NULL
The word in the middle is: in

debugFile from LLMiddleNode_Data1 :

In constructLL method
In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is Four
(dummy, Four) -> NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is score
(dummy, Four) -> (Four, score) -> NULL
In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is and
(dummy, and) -> (and, Four) -> (Four, score) -> NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is seven
(dummy, and) -> (and, Four) -> (Four, seven) -> (seven, score) -> NULL
In listInsert method
Returns from findSpot where Spot.data is score
newNode.data is years
(dummy, and) -> (and, Four) -> (Four, seven) -> (seven, score) -> (score, years) ->
NULL

In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is ago
(dummy, ago) -> (ago, and) -> (and, Four) -> (Four, seven) -> (seven, score) ->
(score, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is our
(dummy, ago) -> (ago, and) -> (and, Four) -> (Four, our) -> (our, seven) ->
(seven, score) -> (score, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is and
newNode.data is fathers
(dummy, ago) -> (ago, and) -> (and, fathers) -> (fathers, Four) -> (Four, our) ->
(our, seven) -> (seven, score) -> (score, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is and
newNode.data is brought
(dummy, ago) -> (ago, and) -> (and, brought) -> (brought, fathers) -> (fathers, Four) ->
(Four, our) -> (our, seven) -> (seven, score) -> (score, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is brought
newNode.data is forth
(dummy, ago) -> (ago, and) -> (and, brought) -> (brought, forth) -> (forth, fathers) ->
(fathers, Four) -> (Four, our) -> (our, seven) -> (seven, score) -> (score, years) ->
NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is on
(dummy, ago) -> (ago, and) -> (and, brought) -> (brought, forth) -> (forth, fathers) ->
(fathers, Four) -> (Four, on) -> (on, our) -> (our, seven) -> (seven, score) ->
(score, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is score
newNode.data is this
(dummy, ago) -> (ago, and) -> (and, brought) -> (brought, forth) -> (forth, fathers) ->
(fathers, Four) -> (Four, on) -> (on, our) -> (our, seven) -> (seven, score) ->
(score, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is brought
newNode.data is continent
(dummy, ago) -> (ago, and) -> (and, brought) -> (brought, continent) -> (continent, forth) ->
(forth, fathers) -> (fathers, Four) -> (Four, on) -> (on, our) -> (our, seven) ->
(seven, score) -> (score, this) -> (this, years) -> NULL

In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is a
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, continent) ->
(continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, on) -> (on, our) ->
(our, seven) -> (seven, score) -> (score, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is new
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, continent) ->
(continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, new) -> (new, on) ->
(on, our) -> (our, seven) -> (seven, score) -> (score, this) -> (this, years) ->
NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is nation
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, continent) ->
(continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, seven) -> (seven, score) -> (score, this) ->
(this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is brought
newNode.data is conceived
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, conceived) ->
(conceived, continent) -> (continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, nation)
->
(nation, new) -> (new, on) -> (on, our) -> (our, seven) -> (seven, score) ->
(score, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is Four
newNode.data is in
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, conceived) ->
(conceived, continent) -> (continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, in) ->
(in, nation) -> (nation, new) -> (new, on) -> (on, our) -> (our, seven) ->
(seven, score) -> (score, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is in
newNode.data is liberty
(dummy, a) -> (a, ago) -> (ago, and) -> (and, brought) -> (brought, conceived) ->
(conceived, continent) -> (continent, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, in) ->
(in, liberty) -> (liberty, nation) -> (nation, new) -> (new, on) -> (on, our) ->
(our, seven) -> (seven, score) -> (score, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is dummy

newNode.data is and
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, forth) -> (forth, fathers) -> (fathers, Four) ->
(Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) -> (new, on) ->
(on, our) -> (our, seven) -> (seven, score) -> (score, this) -> (this, years) ->
NULL
In listInsert method
Returns from findSpot where Spot.data is continent
newNode.data is dedicated
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, dedicated) -> (dedicated, forth) ->
(forth, fathers) ->
(fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, seven) -> (seven, score) -> (score, this) ->
(this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is score
newNode.data is to
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, dedicated) -> (dedicated, forth) ->
(forth, fathers) ->
(fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, seven) -> (seven, score) -> (score, to) ->
(to, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is score
newNode.data is the
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, dedicated) -> (dedicated, forth) ->
(forth, fathers) ->
(fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, seven) -> (seven, score) -> (score, the) ->
(the, to) -> (to, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is our
newNode.data is proposition
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, dedicated) -> (dedicated, forth) ->
(forth, fathers) ->
(fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, proposition) -> (proposition, seven) -> (seven, score) ->
(score, the) -> (the, to) -> (to, this) -> (this, years) -> NULL
In listInsert method

Returns from findSpot where Spot.data is score
newNode.data is that
(dummy, and) -> (and, a) -> (a, ago) -> (ago, and) -> (and, brought) ->
(brought, conceived) -> (conceived, continent) -> (continent, dedicated) -> (dedicated, forth) ->
(forth, fathers) ->
(fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, proposition) -> (proposition, seven) -> (seven, score) ->
(score, that) -> (that, the) -> (the, to) -> (to, this) -> (this, years) ->
NULL
In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is all
(dummy, all) -> (all, and) -> (and, a) -> (a, ago) -> (ago, and) ->
(and, brought) -> (brought, conceived) -> (conceived, continent) -> (continent, dedicated) ->
(dedicated, forth) ->
(forth, fathers) -> (fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, nation) ->
(nation, new) -> (new, on) -> (on, our) -> (our, proposition) -> (proposition, seven) ->
(seven, score) -> (score, that) -> (that, the) -> (the, to) -> (to, this) ->
(this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is liberty
newNode.data is men
(dummy, all) -> (all, and) -> (and, a) -> (a, ago) -> (ago, and) ->
(and, brought) -> (brought, conceived) -> (conceived, continent) -> (continent, dedicated) ->
(dedicated, forth) ->
(forth, fathers) -> (fathers, Four) -> (Four, in) -> (in, liberty) -> (liberty, men) ->
(men, nation) -> (nation, new) -> (new, on) -> (on, our) -> (our, proposition) ->
(proposition, seven) -> (seven, score) -> (score, that) -> (that, the) -> (the, to) ->
(to, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is dummy
newNode.data is are
(dummy, are) -> (are, all) -> (all, and) -> (and, a) -> (a, ago) ->
(ago, and) -> (and, brought) -> (brought, conceived) -> (conceived, continent) -> (continent,
dedicated) ->
(dedicated, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, in) -> (in, liberty) ->
(liberty, men) -> (men, nation) -> (nation, new) -> (new, on) -> (on, our) ->
(our, proposition) -> (proposition, seven) -> (seven, score) -> (score, that) -> (that, the) ->
(the, to) -> (to, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is brought
newNode.data is created
(dummy, are) -> (are, all) -> (all, and) -> (and, a) -> (a, ago) ->

(ago, and) -> (and, brought) -> (brought, created) -> (created, conceived) -> (conceived, continent) ->
(continent, dedicated) -> (dedicated, forth) -> (forth, fathers) -> (fathers, Four) -> (Four, in) ->
(in, liberty) -> (liberty, men) -> (men, nation) -> (nation, new) -> (new, on) ->
(on, our) -> (our, proposition) -> (proposition, seven) -> (seven, score) -> (score, that) ->
(that, the) -> (the, to) -> (to, this) -> (this, years) -> NULL
In listInsert method
Returns from findSpot where Spot.data is dedicated
newNode.data is equal
(dummy, are) -> (are, all) -> (all, and) -> (and, a) -> (a, ago) ->
(ago, and) -> (and, brought) -> (brought, created) -> (created, conceived) -> (conceived, continent) ->
(continent, dedicated) -> (dedicated, equal) -> (equal, forth) -> (forth, fathers) -> (fathers, Four) ->
(Four, in) -> (in, liberty) -> (liberty, men) -> (men, nation) -> (nation, new) ->
(new, on) -> (on, our) -> (our, proposition) -> (proposition, seven) -> (seven, score) ->
(score, that) -> (that, the) -> (the, to) -> (to, this) -> (this, years) ->
NULL
In findMiddleNode method
walker1's data is all
walker1's data is and
walker1's data is a
walker1's data is ago
walker1's data is and
walker1's data is brought
walker1's data is created
walker1's data is conceived
walker1's data is continent
walker1's data is dedicated
walker1's data is equal
walker1's data is forth
walker1's data is fathers
walker1's data is Four
walker1's data is in
In findMiddleNode method
walker1's data is all
walker1's data is and
walker1's data is a
walker1's data is ago
walker1's data is and
walker1's data is brought
walker1's data is created
walker1's data is conceived
walker1's data is continent

walker1's data is dedicated
walker1's data is equal
walker1's data is forth
walker1's data is fathers
walker1's data is Four
walker1's data is in

outFile from LLMiddleNode_Data2 :

(dummy, 84) -> (84, and) -> (and, American) -> (American, about) -> (about, and) ->
(and, apprentice) -> (apprentice, as) -> (as, and) -> (and, a) -> (a, a) ->
(a, and) -> (and, aging) -> (aging, an) -> (an, a) -> (a, and) ->
(and, baseball) -> (baseball, boy) -> (boy, been) -> (been, by) -> (by, been) ->
(been, being) -> (being, between) -> (between, battle) -> (battle, confident) -> (confident, Cuba)
->
(Cuba, catching) -> (catching, day) -> (day, days) -> (days, end) -> (end, each) ->
(each, experienced) -> (experienced, fish) -> (fish, Florida) -> (Florida, far) -> (far, favorite) ->
(favorite, food) -> (food, fishing) -> (fishing, fishermen) -> (fishermen, fish) -> (fish, forbidden) ->
(forbidden, form) -> (form, fish) -> (fish, fisherman) -> (fisherman, Gulf) -> (Gulf, gear) ->
(gear, gone) -> (gone, his) -> (his, he) -> (he, his) -> (his, his) ->
(his, hauling) -> (hauling, has) -> (has, him) -> (him, his) -> (his, has) ->
(has, his) -> (his, He) -> (He, having) -> (having, its) -> (its, is) ->
(is, in) -> (in, into) -> (into, instead) -> (instead, is) -> (is, large) ->
(large, Manolin) -> (Manolin, Manolin) -> (Manolin, marlin) -> (marlin, Man) -> (Man, near) ->
(near, north) -> (north, next) -> (next, night) -> (night, now) -> (now, of) ->
(of, of) -> (of, out) -> (out, on) -> (on, of) -> (of, opens) ->
(opens, of) -> (of, Old) -> (Old, player) -> (player, preparing) -> (preparing, parents) ->
(parents, streak) -> (streak, Straits) -> (Straits, Stream) -> (Stream, Santiago) -> (Santiago,
shack) ->
(shack, Santiagos) -> (Santiagos, successful) -> (successful, sail) -> (sail, so) -> (so, salao) ->
(salao, seen) -> (seen, Santiago) -> (Santiago, story) -> (story, Santiago) -> (Santiago, story) ->
(story, Sea) -> (Sea, that) -> (that, to) -> (to, the) -> (the, the) ->
(the, the) -> (the, that) -> (that, tells) -> (tells, talking) -> (talking, The) ->
(The, to) -> (to, told) -> (told, to) -> (to, that) -> (that, the) ->
(the, The) -> (The, the) -> (the, tells) -> (tells, the) -> (the, The) ->
(The, unlucky) -> (unlucky, unlucky) -> (unlucky, unluckiness) -> (unluckiness, venture) ->
(venture, visits) ->
(visits, will) -> (will, with) -> (with, with) -> (with, worst) -> (worst, without) ->
(without, with) -> (with, young) -> NULL
The word in the middle is: Manolin

debugFile from LLMiddleNode_Data2 :

The debugFile for Data2 contains more than 10 pages.