**CS 700-34 LinkedList Insertion and Find Middle Node**

**Student:** Shuhua Song

**Due Date: Submission Data:**

Soft Copy: 02/04/2020 Soft Copy: 02/04/2020

Hard Copy: 02/06/2020 Hard Copy: 02/06/2020

**Algorithm Steps:**

1. **Main()**

Step 1: Open input file ‘inFile’ by using argv[1]

Open output file ‘outFile1’ and ‘outFile2’ by using argv[2] and argv[3]

Step 2: Create a listHead as the dummy listNode, with value (-9999, nullptr), make the listHead point to the dummy node;

Step 3: Construct a linkedlist in constructLL(inFile, outFile)

Step 4: print the complete ordered final list in printList(outFile)

Step 5: Find the middle node in the ordered linkedlist

Step 6: print the middle node in to the outFile

Step 7: Close all files;

1. Function **constructLL(inFile, outFile)** Steps

Step 1: Read the data from inFile

Step 2: Create a new node by using the new data

Step 3: Insert the new data in the correct position of linkedlist

Step 4: Print the current list for debug purpose

Step 5: repeat step 1-step4 until the end of inFile

1. Function **listInsert(newNode**) steps

Step 1: Find the proper position ‘spot’ in the linkedlist for this new node to insert

Step 2: newNode’s next = spot.next, spot.next = newNode

1. Function **findspot(newNode)** Steps

Step 1: set the spot = listHead

Step 2: if spot’s next != null and spot’s next’s data < newNode’s data

spot = spot’next

Step 3: Repeat step2 until condition failed

Step 4: return spot

1. Function **findMiddleNode(outFile)**

Step 1: set walk1 = listHead, walk2 = listHead

Step 2: printNode(walk1, outFile1)

Step 3: if walk2 != null and walk2’next != null

walk1 = walk1’s next

walk2 = walk2’s next’ next

Step 4: return walk1

*// main.cpp*

*// 700AlgorithmProject1*

*//*

*// Created by Shuhua Song on 2/3/20.*

*// Copyright © 2020 Shuhua Song. All rights reserved.*

*//*

#include <iostream>

#include<string>

#include<fstream>

**using** **namespace** std;

**class** listNode{

**public**:

**int** data;

listNode\* next;

listNode(){

data = -9999;

next = **nullptr**;

}

listNode(**int** data, listNode\* next){

**this**->data = data;

**this**->next = next;

}

**void** printNode(listNode\* node, ofstream& outFile){

*// if(node==nullptr){*

*// outFile << "(" << "NULL" << "," << node << ")->" << endl;*

*// }*

outFile << "(" << node->data << ", " << node

<< ", "<< node->next <<", " << node->next->data << ")" << endl;

}

};

**class** LinkedList{

**public**:

listNode \*listHead;

LinkedList(){

*//creat a dummy node for the linkedlist*

listNode \*dummy = **new** listNode();

listHead = dummy;

}

*//construct a linkedlist*

**void** constructLL(ifstream& inFile, ofstream& outFile){

**if**(inFile.is\_open())

{

**while**(!inFile.eof()){

**int** newData;

inFile >> newData;

**if**(inFile.eof()) **break**;

listNode \*newNode = **new** listNode(newData, **nullptr**);

listInsert(newNode);

printList(outFile);

}

}

**else**{

cout << "Can not open the file" ;

}

}

*//Insert a node after the node of spot*

**void** listInsert(listNode\* newNode){

listNode \*spot = findSpot(newNode);

newNode->next = spot->next;

spot->next = newNode;

}

*//Find a spot that this new node can be inserted after that*

listNode\* findSpot(listNode\* newNode){

listNode \*curNode = listHead;

**while**(curNode->next != **nullptr**){

**if**(curNode->next->data > newNode->data) **break**;

curNode = curNode->next;

}

**return** curNode;

}

**void** printList(ofstream& outFile){

listNode \*curNode = **new** listNode();

curNode = listHead;

outFile << "listHead->";

**while**(curNode->next != **nullptr**){

outFile << "(" << curNode->data << "," << curNode << "," <<

curNode->next << "," << curNode->next->data << ")->" ;

curNode = curNode->next;

}

outFile <<"(" << curNode->data <<"," << "NULL" << ")->";

outFile << "NULL" << endl;

}

*//using fast-slow strategy to find the middle node of linkedlist*

listNode\* findMiddleNode(ofstream& outFile){

listNode\* walk1 = listHead;

listNode\* walk2 = listHead;

listNode\* node = **new** listNode();

**while**(walk2 != **nullptr** && walk2->next != **nullptr** ){

node->printNode(walk1,outFile);

walk1 = walk1->next;

walk2 = walk2->next->next;

}

**return** walk1;

}

};

**int** main(**int** argc, **const** **char** \*argv[]){

*//open and read data from external input file*

ifstream inFile (argv[1]);

*//write the data to output file*

ofstream outFile1 (argv[2]);

ofstream outFile2 (argv[3]);

LinkedList list;

list.constructLL(inFile, outFile2);

outFile1 << "The Completed Sorted Linked list" << endl;

list.printList(outFile1);

listNode\* node = **new** listNode();

listNode\* midNode = list.findMiddleNode(outFile2);

outFile1 << "Middle Node: "<< endl;

node->printNode(midNode,outFile1);

inFile.close();

outFile1.close();

outFile2.close();

**return** 0;

}

The Completed Sorted Linked list

listHead->( 9999,0x7fc50ad00000,0x7fc50ad000b0,8)->(8,0x7fc50ad000b0,0x7fc50ad00150,8)

->(8,0x7fc50ad00150,0x7fc50ad00050,9)->(9,0x7fc50ad00050,0x7fc50ad00070,10)

->(10,0x7fc50ad00070,0x7fc50ad000f0,12)->(12,0x7fc50ad000f0,0x7fc50ad00130,14)

->(14,0x7fc50ad00130,0x7fc50ad00190,29)->(29,0x7fc50ad00190,0x7fc50ad00090,77)

->(77,0x7fc50ad00090,0x7fc50ad001d0,88)->(88,0x7fc50ad001d0,0x7fc50ad00010,91)

->(91,0x7fc50ad00010,0x7fc50ad001b0,91)->(91,0x7fc50ad001b0,0x7fc50ad00250,99)

->(99,0x7fc50ad00250,0x7fc50ad00110,133)->(133,0x7fc50ad00110,0x7fc50ad00030,322)

->(322,0x7fc50ad00030,0x7fc50ad00210,361)->(361,0x7fc50ad00210,0x7fc50ad00170,538)

->(538,0x7fc50ad00170,0x7fc50ad00230,637)->(637,0x7fc50ad00230,0x7fc50ad001f0,702)

->(702,0x7fc50ad001f0,0x7fc50ad000d0,999)->(999,NULL)->NULL

Middle Node:

(91, 0x7fc50ad00010, 0x7fc50ad001b0, 91)