public class Node {

public int iData;

public Node next;

public Node(int id){

iData = id;

}//end constructor

public void displayNode(){

System.out.print("[" + iData + "] -> ");

}//end displayLink

}//end Node

public class LinkedList {

private Node first;

private Node current;

private int size;

public LinkedList(){

first=null;

size = 0;

}

public boolean isEmpty(){

return (first==null);

}

public int **getCount**(int key){

int found = 0;

Node tempCurrent = current; //temp variable to store current

current = first; //start at beginning

while(!(current==null)){

if(current.iData==key){

found += 1; //if found, increment count

current = current.next; //advance pointer

}

else{

current = current.next;

}

}

current = tempCurrent; //restore current

return found;

}

public int **getNth**(int jump){

Node tempFirst = first; //store first

if (!(first == null) && jump < size){

while(jump > 0){ //advance pointer until jump = 0

first = first.next;

jump--;

}//end while

}//end if

else return -1;

Node temp = first; //store found node

first = tempFirst; //reset first

return temp.iData; //return found node

}//end getNth

public void **insertNth**(int index, int key){

Node newNode = new Node(key); //create node

Node tempCurrent = first; //start at front

if (index <= size - 1){

for(int i=index-1; i>0; i--){ //traverse index-1 amount of times

tempCurrent = tempCurrent.next;

}//end for

}//end if

newNode.next = tempCurrent.next;

tempCurrent.next = newNode;

size++;

}//end insertNth

public void **insertEnd**(int id){

Node newNode = new Node(id);

if(isEmpty()){

first = newNode;

current = newNode;

current.next = null;

}//end if

else{

current.next = newNode;

current = newNode;

current.next = null;

}//end else

size++;

}//end insertEnd

public void **reverse**(){

if(size<=1) return; //if size<=1, nothing needs to reverse

else{

Node nearNode = first; //near node starts at beginning

Node midNode;

Node farNode;

midNode = nearNode.next; //second in line

farNode = midNode.next; //next over from midNode

nearNode.next = null;

while(farNode!=null){ //until far reaches null

midNode.next = nearNode;

nearNode = midNode; //swap near to mid

midNode = farNode; //swap mid to far

farNode = farNode.next; //advance far

}//end while

midNode.next = nearNode;

first = midNode;

}//end else

}//end reverse()

public void displayList(){

System.out.print("List (first to last): ");

Node current = first;

while(current != null){

current.displayNode();

current = current.next;

}

System.out.println("");

}//end displayLIst()

}//end LinkedList

public class LinkListApp {

public static void main(String[] args) {

LinkedList list = new LinkedList();

list.insertEnd(1);

list.insertEnd(2);

list.insertEnd(3);

list.insertEnd(4);

list.insertEnd(4);

list.insertEnd(4);

list.displayList();

list.insertNth(5, 15);

list.insertEnd(3);

list.displayList();

System.out.println("Data found: " + list.getNth(0));

System.out.println("Key found: " + list.getCount(5) + " times.");

System.out.println("Key found: " + list.getCount(3) + " times.");

list.reverse();

list.displayList();

}}//outputs:

List (first to last): [1] -> [2] -> [3] -> [4] -> [4] -> [4] ->

List (first to last): [1] -> [2] -> [3] -> [4] -> [4] -> [15] -> [4] -> [3] ->

Data found: 4

Key found: 0 times.

Key found: 2 times.

List (first to last): [3] -> [4] -> [15] -> [4] -> [4] -> [3] -> [2] -> [1] ->