1)

package javaapplication60;

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public boolean equivalents(SinglyLinkedList otherList) {

Node current1 = this.head;

Node current2 = otherList.head;

while (current1 != null && current2 != null) {

if (current1.data != current2.data) {

return false;

}

current1 = current1.next;

current2 = current2.next;

}

return current1 == null && current2 == null;

}

}

public class JavaApplication60 {

public static void main(String[] args) {

SinglyLinkedList list1 = new SinglyLinkedList();

SinglyLinkedList list2 = new SinglyLinkedList();

list1.append(1);

list1.append(2);

list1.append(3);

list2.append(1);

list2.append(2);

list2.append(3);

System.out.println("Are lists equivalent? " + list1.equivalents(list2));

list2.append(4);

System.out.println("Are lists equivalent? " + list1.equivalents(list2e

}

}

2)

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public Node findSecondToLastNode() {

if (head == null || head.next == null) {

return null;

}

Node current = head;

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

current = current.next;

}

return current;

}

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.append(10);

list.append(20);

list.append(30);

list.append(40);

Node secondToLast = list.findSecondToLastNode();

if (secondToLast != null) {

System.out.println("Second to last node is: " + secondToLast.data);

} else {

System.out.println("No second to last node exists.");

}

}

}

3)

package javaapplication60;

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public int size() {

int count = 0;

Node current = head;

while (current != null) {

count++;

current = current.next;

}

return count;

}

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.append(10);

list.append(20);

list.append(30);

System.out.println("حجم القائمة: " + list.size());

}

}

4)

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void rotate() {

if (head == null || head.next == null) {

return;

}

Node oldHead = head;

head = head.next;

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = oldHead;

oldHead.next = null;

}

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.append(10);

list.append(20);

list.append(30);

list.append(40);

System.out.println("القائمة الأصلية:");

list.printList();

list.rotate();

System.out.println("بعد التدوير الأول:");

list.printList();

list.rotate();

System.out.println("بعد التدوير الثاني:");

list.printList();

}

4)

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void rotate() {

if (head == null || head.next == null) {

return;

}

Node oldHead = head;

head = head.next;

Node current = head;

while (current.next != null) {

current = current.next; // الوصول إلى العقدة الأخيرة

}

current.next = oldHead; // ربط العقدة الأخيرة بالرأس القديم

oldHead.next = null; // جعل الرأس القديم يشير إلى null

}

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.append(10);

list.append(20);

list.append(30);

list.append(40);

System.out.println("القائمة الأصلية:");

list.printList();

list.rotate();

System.out.println("بعد التدوير الأول:");

list.printList();

list.rotate();

System.out.println("بعد التدوير الثاني:");

list.printList();

}

}

5)

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void concatenate(SinglyLinkedList otherList) {

if (head == null) {

head = otherList.head;

return;

}

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = otherList.head;

}

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public static void main(String[] args) {

SinglyLinkedList list1 = new SinglyLinkedList();

SinglyLinkedList list2 = new SinglyLinkedList();

list1.append(1);

list1.append(2);

list1.append(3);

list2.append(4);

list2.append(5);

list2.append(6);

System.out.println("List 1:");

list1.printList();

System.out.println("List 2:");

list2.printList();

list1.concatenate(list2);

System.out.println("Concatenated List:");

list1.printList();

}

}

6)

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

// Append a new node to the list

public void append(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void reverse() {

Node prev = null;

Node current = head;

while (current != null) {

Node next = current.next; // Save the next node

current.next = prev; // Reverse the pointer

prev = current; // Move prev forward

current = next; // Move current forward

}

head = prev;

}

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public static void main(String[] args) {

SinglyLinkedList list = new SinglyLinkedList();

list.append(1);

list.append(2);

list.append(3);

list.append(4);

System.out.println("Original List:");

list.printList();

list.reverse();

System.out.println("Reversed List:");

list.printList();

}

}