**Data Structure Lab 4**

**Salsabil AL -Warraq**

Q2 : Give an algorithm for finding the second-to-last node in a singly linked list in which the last node is indicated by a null next reference

الخوارزمية لإيجاد العقدة قبل الأخيرة:

الحالة الخاصة: إذا كانت القائمة تحتوي على أقل من عقدتين، ارجع بـ null (لأنه لا توجد عقدة قبل الأخيرة).

قم بتهيئة مؤشرين:

current: يشير إلى رأس القائمة.

previous: يشير في البداية إلى null.

قم بالتكرار عبر القائمة باستخدام current:

إذا كان current.next == null، فهذا يعني أن previous هو العقدة قبل الأخيرة.

حرك previous ليشير إلى current، و current ليشير إلى current.next.

إذا وصلت إلى نهاية القائمة، وأصبحت current.next == null، فإن previous ستكون العقدة قبل الأخيرة.

إذا لم تكن هناك عقدة قبل الأخيرة (أي القائمة تحتوي على أقل من عقدتين)، ارجع بـ null.

الكود:

**public Node<E> findSecondToLast( ) {**

**if (head == null || head.getNext() == null) {**

**return null; // إذا كانت القائمة تحتوي على أقل من عقدتين**

**}**

**Node<E> current = head;**

**Node<E> previous = null;**

**// التكرار عبر القائمة**

**while (current != null && current.getNext() != null) {**

**previous = current;**

**current = current.getNext();**

**}**

**return previous; // إرجاع العقدة قبل الأخيرة}**

**{**

Q3 : Give an implementation of the size( ) method for the SingularlyLinkedList class, assuming that we did not maintain size as an instance variable

public int size() {

int count = 0;

Node<E> current = head;

while (current != null) {

count++;

current = current.getNext();

}

return count;

}

Q5 : Describe an algorithm for concatenating two singly linked lists L and M, into a single list L′ that contains all the nodes of L followed by all the nodes of M.

**خطوات الخوارزمية:**

1. **التحقق من حالة القائمتين**:
   * إذا كانت L فارغة، فإن L' ستصبح ببساطة M.
   * إذا كانت M فارغة، فإن L' ستظل كما هي.
2. **جمع العناصر**:
   * إذا كانت القائمتان تحتويان على عناصر، قم بالبحث عن نهاية L ( العقدة الأخيرة).
   * قم بتوصيل العقدة الأخيرة في L برأس M.
3. **تحديث المؤشرات**:
   * تأكد من أن tail في L' يشير الآن إلى tail في M

public void concatenate(SinglyLinkedList<E> M) {

if (this.isEmpty()) {

this.head = M.head;

this.tail = M.tail;

} else if (!M.isEmpty())

this.tail.setNext(M.head);

this.tail = M.tail;

}

{

Q6: Describe in detail an algorithm for reversing a singly linked list L using only a constant amount of additional space.

عكس القائمة المرتبطة المفردة يتضمن إعادة ترتيب المؤشرات للعقد بحيث تشير إلى الاتجاه المعاكس. تستخدم فقط كمية ثابتة من المساحة الإضافية

### خطوات الخوارزمية:

1. **تهيئة المؤشرات**:
   * قم بإنشاء ثلاثة مؤشرات:
     + prev مُهيأ إلى null.
     + current مُهيأ إلى head (رأس القائمة).
     + next لتخزين العقدة التالية بشكل مؤقت.
2. **التكرار عبر القائمة**:
   * بينما يكون current ليس null:
     + **تخزين العقدة التالية**:
       - قم بتعيين next إلى current.next (هذا يخزن العقدة التالية قبل تغييرها).
     + **عكس المؤشر**:
       - قم بتعيين current.next إلى prev (هذا يعكس الرابط).
     + **تحريك المؤشرات للأمام**:
       - قم بتحريك prev إلى current (نقل prev إلى العقدة الحالية).
       - قم بتحريك current إلى next (نقل current إلى العقدة التالية المخزنة سابقًا).
3. **تحديث الرأس**:
   * بعد انتهاء الحلقة، قم بتعيين head للقائمة إلى prev (الذي سيكون الآن الرأس الجديد للقائمة المعكوسة)
   * )