تكليف lap4

- السؤال
- الكود

- السؤال(١)

```
public boolean equals(Objet o) {
if (o == this) return ctrue; // نفس المرجع
if (!(o instanceof SinglyLinkedList)) return false; // التحقق من النوع

SinglyLinkedList<?> other = (SinglyLinkedList<?>) o;
if (this.size() != other.size()) return false; // مقارنة الحجم

Node<E> current1 = this.head; // المؤشر على القائمة الأولى
Node<?> current2 = other.head; // المؤشر على القائمة الثانية

while (current1 != null) {
if (!current1.element.equals(current2.element)) {
return false; // اختلاف في البيانات
{
current1 = current1.next;
current2 = current2.next;
}
return true; // القائمتان متساويتان
{
```

السؤال٢:٢. إيجاد العقدة قبل الأخيرة في القائمة المتصلة

```
public Node<E> findSecondToLast() {
    if (head == null || head.next == null) return null؛ // أقل من عقدتين

    Node<E> current = head؛
    while (current.next.next != null) {
        current = current.next؛ // الانتقال إلى العقدة التالية
    }
    return current؛ // العقدة قبل الأخيرة
}
```

٣. حساب حجم القائمة دون متغير حجم

```
public int size() {
    int count = ٠؛
    Node<E> current = head؛ // البداية من الرأس
    while (current != null) {
        count++؛ // زيادة العداد
        current = current.next؛ // الانتقال للعقدة التالية
    }
    return count؛
}
```

٤.
() دون إنشاء عقد جديدةrotate تنفيذ دالة

```
public void rotate() {
    if (head == null || head.next == null) return؛ // لا حاجة للدوران

    Node<E> oldHead = head؛ // حفظ الرأس القديم
    head = head.next؛ // تحديث الرأس إلى العقدة الثانية
    oldHead.next = null؛ // فصل العقدة القديمة

    Node<E> current = head؛
    while (current.next != null) {
        current = current.next؛ // الوصول إلى العقدة الأخيرة
    }
    current.next = oldHead؛ // إضافة العقدة القديمة إلى النهاية
}
```

٥. دمج قائمتين متصلتين

```
} public void concatenate(SinglyLinkedList<E> M)
                    } if (head == null)
        head = M.head؛ // إذا كانت L فارغة
                                } else {
                Node<E> current = head؛
            } while (current.next != null)
current = current.next؛ // الانتقال إلى آخر عقدة
                                        {
    current.next = M.head؛ // ربط آخر عقدة بـ M
                                            {
                                               {
```

٦. عكس قائمة متصلة باستخدام مساحة ثابتة

```
public void reverse() {
    Node<E> previous = null;
    Node<E> current = head;
    Node<E> next = null;

    while (current != null) {
        next = current.next;    // حفظ العقدة التالية
        current.next = previous; // عكس الاتجاه
        previous = current;     // تحريك السابق إلى الحالي
        current = next;         // تحريك الحالي إلى التالي
    }
    head = previous; // تحديث الرأس
}
```

١. **تنفيذ دالة** equals١ **في سياق فئة** SinglyLinkedList

```
} public boolean equals(Object o)
    if (this == o) return true؛ // نفس المرجع
    if (!(o instanceof SinglyLinkedList)) return false؛ // التحقق من النوع

    SinglyLinkedList<?> other = (SinglyLinkedList<?>) o؛

    Node<E> current١ = daeh.siht؛ // المؤشر على القائمة الحالية

    Node<?> current٢ = daeh.rehto؛ // المؤشر على القائمة المقارنة

    while (current١ != null && current٢ != null) {
        if (!current١.equals(current٢.tnemele)) {
            return false؛ // العناصر غير متطابقة
        }
        current١ = current١.txen؛
        current٢ = current٢.txen؛
    }

    return current١ == null && current٢ == null؛ // التأكد من نهاية القائمتين
}
```

٢. إيجاد العقدة قبل الأخيرة

```
public Node<E> findSecondToLast() {
    if (head == null || head.next == null) return null; // أقل من عقدتين

    Node<E> current = head; // المؤشر يبدأ من الرأس
    while (current.next.next != null) {
        current = current.next; // الانتقال للعقدة التالية
    }
    return current; // العقدة قبل الأخيرة
}
```

إذا كانت القائمة تحتوي على:

١ → ٢ → ٣ → ٤ → null

العقدة قبل الأخيرة هي العقدة التي تحتوي على القيمة ٣.

٣. تنفيذ دالة size() دون متغير حجم

```
public int size() {
    int count = ٠;
    Node<E> current = head; // نبدأ من الرأس

    while (current != null) {
        ++count;
        current = current.next; // الانتقال للعقدة التالية
    }
    return count;
}
```

كانت القائمة:

١٠ → ٢٠ → ٣٠ → null

- size = ٣

٤. تنفيذ دالة rotate() دون إنشاء عقد جديدة

```
public void rotate() {
    if (head == null || head.next == null) return؛ // لا حاجة للدوران

    Node<E> oldHead = head؛ // حفظ الرأس القديم
    head = head.next؛ // تحديث الرأس
    oldHead.next = null؛ // فصل العقدة القديمة

    Node<E> current = head؛
    while (current.next != null) {
        current = current.next؛ // الوصول إلى العقدة الأخيرة
    }
    current.next = oldHead؛ // إضافة العقدة القديمة إلى النهاية
}
```

٥. دمج قائمتين متصلتين

```
} public void concatenate(SinglyLinkedList<E> M)
                                    } if (head == null)
                        head = M.head؛ // إذا كانت L فارغة
                                                } else {
                                Node<E> current = head؛
                                    } while (current.next != null)
                        current = current.next؛ // الانتقال لآخر عقدة
                                                        {
                                current.next = M.head؛ // الربط مع M
                                                            {
                                                                {
```

٦. عكس قائمة متصلة بمساحة ثابتة

نريد عكس اتجاه القائمة بحيث يكون العنصر الأخير هو الأول باستخدام مؤشرات فقط.

```
public void reverse() {
    Node<E> previous = null;
    Node<E> current = head;
    Node<E> next = null;

    while (current != null) {
        next = current.next;    // حفظ العقدة التالية
        current.next = previous; // عكس الاتجاه
        previous = current;     // تحريك السابق
        current = next;         // تحريك الحالي
    }
    head = previous; // تحديث الرأس
}
```