Linked List Assignment Solutions حل واجب القائمة المترابطة

1.خطأ في التعريف(Error in the definition)

- **الخطأ (Error)**: `Arrays arrays = new Arrays();`
- **التصحيح (Correction)**: لا يمكن إنشاء كائن من `Arrays` لأنها فئة أدوات. يمكن استخدام الطرق مثل `Arrays.sort(array)` مباشرة.

2.طريقة لعكس مصفوفة(Reverse method for an array)

Code الكود:

```
void reverse(int[] a) {
    int n = a.length;
for (int i = 0; i < n / 2; i++) {
     int temp = a[i];
    a[i] = a[n - i - 1];
    a[n - i - 1] = temp;
                }
                }
```

Explanation شرح : يتم تبديل العناصر من البداية والنهاية حتى منتصف المصفوفة.

3.مقارنة بين المصفوفات والقوائم المترابطة(Arrays vs Linked Lists)

سبب استخدام المصفوفات:(Why arrays are used)

1. **الوصول العشوائي :(Random access)** يمكن الوصول إلى العناصر مباشرة باستخدام الفهرس.
2. **الموقع في الذاكرة :(Memory locality)** البيانات متجاورة في الذاكرة، مما يحسن الأداء.
3. **البساطة :(Simplicity)** سهلة الاستخدام للبيانات ذات الحجم الثابت.

4.العبارات صحيحة أم خاطئة(True/False Statements)

- **أ :(a)** صحيح(True)
- **ب :(b)** خطأ(False)
- **ج :(c)** خطأ(False)
- **د :(d)** صحيح(True)
- **هـ :(e)** خطأ(False)

5.نواتج عبارات الجافا(Outputs of Java Statements)

- **أ :(a)** قيمة العنصر في أول عقدة.
- **ب :(b)** قيمة العنصر في العقدة .A
- **ج :(c)** قيمة العنصر بعد العقدة .B
- **د :(d)** قيمة العنصر في العقدة الثالثة.

6.التعبيرات المنطقية(Relational Expressions)

- **أ :(a)** يعتمد على قيمة .`list.getElement()`
- **ب :(b)** صحيح أو خطأ بناءً على ما إذا كانت العقدة الثانية هي .A
- **ج :(c)** صحيح إذا كانت القيمة في العقدة التالية لـ Aتساوي 16.
- **د :(d)** صحيح إذا كانت Bهي العقدة الأخيرة.
- **هـ :(e)** صحيح أو خطأ بناءً على قيمة العنصر الأول.

7.شفرات جافا المطلوبة(Java Code Fragments)

- أ; **(a):** `A = findNode(23);`
- ب; **(b):** `list = findNode(16);`
- ج; **(c):** `B = findLastNode();`
- د; **(d):** `list = null;`
- هـ; **(e):** `findNode(25).setElement(35);`
- و **(f):** إضافة بعد A:

```
Node newNode = new Node(10);
newNode.setNext(A.getNext());
A.setNext(newNode);
```

- ز **(g):** حذف العقدة 23:

```
Node prev = findPrevious(23);
Node toDelete = prev.getNext();
prev.setNext(toDelete.getNext());
toDelete = null; // تحرير الذاكرة
```

8.مخرجات الكود(Output of Java Code)

الوصف :طباعة جميع عناصر القائمة بالتسلسل.

9.مخرجات الشفرات المعطاة(Outputs of Given Java Code)

- أ **(a):** طباعة.10, 18, 13 :
- ب **(b):** طباعة.30, 42, 28, 20 :

10.مخرجات البرنامج(Program Output)

الناتج.`{18, 38, 15, 45, 25}` **(Output):**

11.إدراج 20 بين 15 و24(Insert 20 between 15 and 24)

```
Node newNode = new Node(20);
newNode.setNext(node15.getNext());
node15.setNext(newNode);
```

12.جمع عناصر القائمة(Sum Method)

الكود:(Code)

```
public int sum(Node<Integer> list) {
                        int total = 0;
                while (list != null) {
            total += list.getElement();
                 list = list.getNext();
                                }
                          return total;
                                }
```

13.حذف العقدة الأخيرة(Remove Last Node)

الكود:(Code)

```
 public E removeLast(Node<E> list) {
Node<E> prev = null, current = list;
 while (current.getNext() != null) {
                        prev = current;
           current = current.getNext();
                                }
                    prev.setNext(null);
           return current.getElement();
                                }
```

14.إلحاق قائمتين(Append Two Lists)

الكود:(Code)

```
public void append(Node<E> list1, Node<E> list2) {
                                Node<E> temp = list1;
                    while (temp.getNext() != null) {
                                temp = temp.getNext();
                                                    }
                                  temp.setNext(list2);
                                                    }
```

15.دمج قائمتين(Concatenate Two Lists)

الكود:(Code)

```
public Node<E> concat(Node<E> list1, Node<E> list2) {
                    Node<E> newHead = copyList(list1);
                              Node<E> temp = newHead;
                    while (temp.getNext() != null) {
                                temp = temp.getNext();
                                                    }
                      temp.setNext(copyList(list2));
                                        return newHead;
                                                    }
```

16.تبديل العناصر(Swap Elements)

الكود:(Code)

```
        public void swap(Node<E> list, int i, int j) {
                    Node<E> nodeI = getNodeAt(list, i);
                    Node<E> nodeJ = getNodeAt(list, j);
                           E temp = nodeI.getElement();
                 nodeI.setElement(nodeJ.getElement());
                                nodeJ.setElement(temp);
                                                    }
```

17.عكس قائمة مفردة(Reverse Algorithm)

1. **بدءياً.(Initialize):** `prev = null, current = head`
2. **التكرار :(Iterate)**
   - حفظ. `next = current.next`
   - تعيين. `current.next = prev`
   - تحديث. `prev = current, current = next`:
3. **تحديث الرأس.(Update head):** `head = prev`

18.تنفيذ(Implement equals()) equals()

الوصف :مقارنة القوائم عقدة بعقدة.

19.تنفيذ(Implement rotate()) rotate()

الوصف :نقل العقدة الرأسية إلى نهاية القائمة الدائرية.

20.تنفيذ(Implement addFirst()) addFirst()

الوصف :إضافة عقدة جديدة في بداية القائمة الدائرية.