1- What's wrong with this definition:

```
Arrays arrays = new Arrays();

int[] numbers = new int[5];
```

2- Write and test this method:

```
void reverse(int[] a)
```

```
public class ReverseArray {
   public static void main(String[] args) {
      int[] array = {1, 2, 3, 4, 5};

      System.out.println("Original array:");
      printArray(array);

      reverse(array);

      System.out.println("\nReversed array:");
      printArray(array);
   }

   static void reverse(int[] a) {
      int start = 0;
      int end = a.length - 1;

      while (start < end) {
         int temp = a[start];
         a[start] = a[end];
         a[end] = temp;

         start++;
         end--;
      }
   }

   static void printArray(int[] a) {
      for (int i = 0; i < a.length; i++) {
         System.out.print(a[i] + " ");
      }
   }
}
```

3- If linked lists are so much better than arrays, why are arrays used at all?

1. سهولة الوصول (index).

2. توفير الذاكرة.

3. إمكانية التعديل.

4. الاستخدامات القائمة على الفهرس
5. الاستخدامات المحدودة للقوائم المرتبطة

4- What is the output of each of the following java statements?

1. System.out.println(list.getElement()): هنا، نفترض أن list هو قائمة مرتبطة ولديها طريقة getElement() التي تعيد قيمة العنصر الحالي في القائمة. ستتم طباعة قيمة العنصر الحالي في list.

2. System.out.println(A.getElement()): هنا، نفترض أن A هو كائن من نوع يحتوي على طريقة getElement() التي تعيد قيمة محددة. ستتم طباعة القيمة المحددة.

3. System.out.println(B.getNext().getElement()): هنا، نفترض أن B هو كائن من نوع يحتوي على طريقة getNext() التي تعيد العنصر التالي في القائمة. سيتم استدعاء طريقة getElement() على العنصر التالي وسيتم طباعة القيمة المحددة.

4. System.out.println(list.getNext().getNext().getElement()): هنا، نفترض أن list هو قائمة مرتبطة ولديها طريقة getNext() التي تعيد العنصر التالي في القائمة. سيتم استدعاء getNext() مرتين، ثم سيتم استدعاء getElement() على العنصر النهائي وسيتم طباعة القيمة المحددة.

1. list.getElement() >= 18: هذه التعبيرة تحقق ما إذا كانت قيمة العنصر الحالي في list أكبر من أو تساوي 18. إذا كانت القيمة أكبر من أو تساوي 18، فإن التعبير سيكون صحيحًا (true)، وإلا فسيكون غير صحيح (false).

2. list.getNext() == A: هذه التعبيرة تحقق ما إذا كان العنصر التالي في list يشير إلى نفس العنصر التالي الذي يتبعه A. إذا كان A هو نفسه الذي يشير إليه list، فإن التعبير سيكون صحيحًا (true)، وإلا فسيكون غير صحيح (false).

3. A.getNext().getElement() == 16: هذه التعبيرة تحقق ما إذا كانت قيمة العنصر التالي بعد A تساوي 16. إذا كانت القيمة تساوي 16، فإن التعبير سيكون صحيحًا (true)، وإلا فسيكون غير صحيح (false).

4. B.getNext() == (NULL): هذه التعبيرة تحقق ما إذا كان العنصر التالي بعد B يشير إلى قيمة NULL، وهي قيمة تشير إلى عدم وجود عنصر معين. إذا كان العنصر التالي بعد B يشير إلى NULL، فإن التعبير سيكون صحيحًا (true)، وإلا فسيكون غير صحيح (false).

5. list.getElement() == 18: هذه التعبيرة تحقق ما إذا كانت قيمة العنصر الحالي في list تساوي 18. إذا كانت القيمة تساوي 18، فإن التعبير سيكون صحيحًا (true)، وإلا فسيكون غير صحيح (false).

7-Write java Fragment code to do the following:

Make A point to the node containing element 23.
Make list point to the node containing 16.Make B point to the last node in the list.

Make list point to an empty list.
Set the value of the node containing 25 to 35.
Create and insert the node with element 10 after the node pointed by A.
Delete the node with element 23. Also, deallocate the memory occupied by this node.

```
class Node {
   int element;
   Node next;

   Node(int element) {
      this.element = element;
      this.next = null;
   }
}
public class LinkedListOperations {
   public static void main(String[] args) {
      Node A = null;
      Node list = null;
      Node B = null;
            Node newNode1 = new Node(23);
      A = newNode1;
Node newNode2 = new Node(16);
list = newNode2;
     B = list;
    list = null;

if (newNode1 != null && newNode1.element == 25) {
   newNode1.element = 35;
}

Node newNode3 = new Node(10);
if (A != null) {
   newNode3.next = A.next;
   A.next = newNode3;
}

if (list != null && list.element == 23) {
   list = list.next;
}
}
}
```

5- What is the output of the following java code?

```
p = list;
while (p != NULL){
System.out.println(  p.getElement());
p = p.getNext();  }
```

نفيذ الخطوات التالية:

1. p = list;: يتم تعيين المؤشر p ليشير إلى نفس العنصر الذي يشير إليه list.

2. يتم تكرار الكود داخل الحلقة while (عندما يصبح p يساوي NULL).

3. باستخدام System.out.println(p.getElement()) في كل تكرار، يتم طباعة قيمة العنصر الحالي الذي يشير إليه p.

4. p = p.getNext(). ليشير إلى العنصر التالي في القائمة باستخدام p بعد ذلك، يتم تحديث

النتيجة المتوقعة هي طباعة قيمة كل عنصر في القائمة المرتبطة على سطر منفصل. وفي نهاية القائمة، عندما يصبح p يساوي NULL، تتوقف الحلقة ويتم الخروج منها.

9-Write and test this method for SingleLinkedList class :

Public int sum(Node<int> list)

For example, if list is {25, 45, 65, 85}, then sum(list) will return 220

```
class Node<T> {
    T element;
    Node<T> next;

    Node(T element) {
      this.element = element;
      this.next = null;
    }
  }

  class SingleLinkedList {
    public static int sum(Node<Integer> list) {
      int sum = 0;
      Node<Integer> current = list;

      while (current != null) {
        sum += current.element;
        current = current.next;
      }

 return sum;
```

```
    }

    public static void main(String[] args) {
        Node<Integer> list = new Node<>(25);
        list.next = new Node<>(45);
        list.next.next = new Node<>(65);
        list.next.next.next = new Node<>(85);

        int result = sum(list);
        System.out.println("Sum: " + result); // Output: Sum: 220
    }
}
```