Q6: Array Programming

int[]

Write a static method called insertMiddle that takes in two integer arrays as parameters
(we'll call them arr1 and arr2) and returns a new array that contains:
1. the elements from the first half of arr1   int[]
2. then, all of the elements of arr2
3. finally, the rest of the elements in the second half of arr1

For example, consider the following two arrays:

[1, 2, 3, 4, 5, 6]

```
int[] arr1 = {1, 2, 5, 6};
int[] arr2 = {3, 4};
```

A call to insertMiddle(arr1, arr2) should return the following array:

```
{1, 2, 3, 4, 5, 6}
```

It may be the case that the first array has an odd length. In this case, treat the first half as
the "shorter half" of the two. For example, consider the following two arrays:

```
int[] arr3 = {2, 4, 6, 8, 10};
int[] arr4 = {1, 1, 1};
```

A call to insertMiddle(arr3, arr4) should return the following array:

```
{2, 4, 1, 1, 1, 6, 8, 10}
```

In addition,
- you may assume that neither array is null
- you may not assume that neither array is non-empty; in other words, your method
  should also work if one (or both) of the arrays has length 0

As a reminder, you are restricted to the methods and classes provided on the reference
sheet. Your method should not modify either of the arrays that are provided as
parameters.

```java
public static int[] insertMiddle
        (int[] arr1, int[] arr2) {

    int[] result = new int[arr1.length +
                            arr2.length];

    for (int i = 0; i < arr1.length/2; i++) {

        result[i] = arr1[i];

    }

    for (int i = 0; i < arr2.length; i++) {
            i + arr1.length/2
        result[i] = arr2[i];
            result(2) = arr2(0)
    }

    for (int i = arr1.length/2;
            i < arr1.length; i++) {

        result[] = arr1[];

    }

    return result;
```
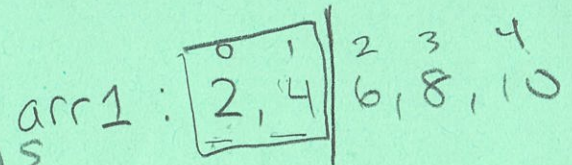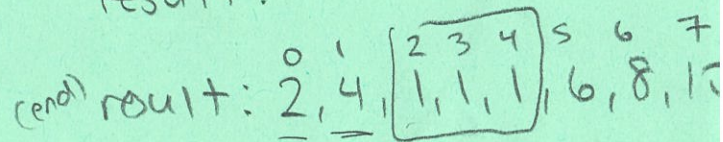
- Create new array (return)
- For loop to traverse arrays
  - Traverse 1st half of arr1 and fill in new arr
  - Traverse entirity of arr2 and fill new array
  - Traverse 2nd half of arr1 and fill in
- Return new array

arr1 :

0	1	2	3	4
2	4	6	8	10

arr 2 :

0	1	2
1	1	1

result:

(end) result:

0	1	2	3	4	5	6	7
2	4	1	1	1	6	8	10

3