U6b Review Lab:
public static int[] rightDigitDuplicator(int[] nums) + Algorithms for Arrays implementation

{SUGGESTED STRATEGY: }

- Handwrite your code on a blank sheet of paper to mimic and practice FRQ exam responses.
- Use your handwritten code from which to work making additions, deletions, corrections as your syntax is developed into executable source code that satisfies the given post condition.
- Include a PMR (post-mortem review) in the project header noting your edits from FRQ to source.
- After completion of this lab, navigate to AP Central CSA The Exam page reviewing FRQ 2017#1 part a noticing use of modulus (%) to identify and duplicate the rightmost digit of a number sequence (aka "digit stripping"). Another example of a similar use of modulus is seen in CED Sample FRQ #1, checkDigit, parts a and b. Locate this FRQ within the CSA CED on pages 199-202, and see your course instructor for review of the canonical solution. IOW, analyze the FRQ and apply your knowledge to prepare for similar application of Java solutions.
- Refer to CSA CED CON-2.I.1 or Topic 6.4 on page 116 for the standard algorithms that utilize array traversals. This lab requires several of those algorithms. Students should be prepared to implement each of the algorithms listed.

==

PRE-CONDITION:

An `int[]` array, `nums`, contains *n* random `int` objects >= 0.

POST-CONDITION:

USE MODULUS TO IDENTIFY RIGHTMOST DIGIT :

1. `public static int[] rightDigitDuplicator(int[] nums)` creates a new array, `rightNums[]`, containing the right most digit of all integers in `nums[]`, returns `rightNums[]`.

USE ALGORITHMS THAT UTILIZE ARRAY TRAVERSAL :

2. `public static int[] evenNumsArray (int[] array)` takes an array of integers; creates a new array, `evenNums`; body of the method populates `evenNums` with all even numbered values contained in the traversed array.
3. `public static int maxNum (int[] array)` takes an array of integers, returns the maximum value contained in the array.
4. `public static boolean findUpperRange (int[]array, int upperRange)` takes an array of integers and its upper range bound, returns `true` if

the upper boundary value is present in the array. Example: If the range of random values that filled the array is 1-100, is 100 present in the array.

5. `public static int countConsecutivePairs(int[] array)` takes an array of integers, returns the count of all consecutive pairs of same integer elements in the array.
6. `public static int indexOfConsecutivePairs(int[] array)` takes an array of integers, returns the index value(s) of the first of each consecutive pair found in the array or -1 if no consecutive pairs are present in the array.

SUBMIT:

Java source code showing:

- declare, populate three `int[]` arrays, `numsA-C`, as detailed below:

	Length of `nums[]`	Range of random values (inclusive)
`numsA`	25	10 - 999
`numsB`	50	1,000 - 99,999
`numsC`	100	1 - 2,021

  - call to `rightDigitDuplicator(numsA[]);` returns `rightNumsA[]`
  - call to `rightDigitDuplicator(numsB[]);` returns `rightNumsB[]`
  - call to `rightDigitDuplicator(numsC[]);` returns `rightNumsC[]`

- Java source code for *at least* one test case each for the algorithm traversals methods:

```
  i.    evenNumsArray(numsA[])
 ii.    maxNum(evenNumsA[]) //param is resulting array from traversal i. above
iii.    findUpperRange(numsB[])
 iv.    countConsecutivePairs(numsC[])
  v.    indexOfConsecutivePairs(numsC[])
```

  ✓ NOTE: use comments to provide heading detail for the above traversals

Lab detail:

- 6 methods created
- 3 arrays created using Math.random()
- 8 method calls