Adjoining Digits

You will implement three static methods in the `AdjoiningDigits` class. The three methods you will implement are the `isDivisibleBy(int number, int[] divisors)`, the `getLCM(int[] num)` and the `adjoinDigits(int num, int numDigits, int[] divisors)` methods.

The `isDivisibleBy(int number, int[] divisors)` method returns true if every element of the `int[]` parameter `divisors` divides into the parameter `number`.
The following code shows the results of the `isDivisibleBy` method.

The following code	Returns

`int[] divisors = new int[] {2, 5, 50};`
 	
`AdjoiningDigits.isDivisibleBy(100, divisors);`
	`true`

`int[] divisors = new int[] {2, 5, 30};`
 	
`AdjoiningDigits.isDivisibleBy(100, divisors);`
	`false`

The `getLCM(int[]num)` method determines the Least Common Multiple of every element in the `int[]` parameter `num`.

The following code shows the results of the `getLCM` method.

The following code	Returns

`int[] num = new int[] {2, 5, 50};`
 	
`AdjoiningDigits.getLCM(num);`
	`50`

`int[] num = new int[] {2, 5, 15, 30};`
 	
`AdjoiningDigits.getLCM(num);`
	`30`

`int[] num = new int[] {3, 5, 7};`
 	
`AdjoiningDigits.getLCM(num);`
	`105`

Problem continues on next page

The motivation for the `adjoinDigits()` comes from the following problem:

Adjoin to the digits 523, a total of **exactly** three digits such that the resulting 6 digit number is divisibile by 7, 8 and 9.

Note: adding leading zeros is NOT considered adjoining digits. Furthermore, the number 012 is consider a 2 digit number

The `adjoinDigits(int num, int numDigits, int[] divisors)` returns the smallest `int` value, `answer`, such that:

- `answer` contains the digits from `num`.
  Or, in code:
  `("" + answer).indexOf("" + num) >= 0;`
- **Exactly** `numDigits` digits have been adjoined to num. All, some or no digits may be added to the front of `num` and all, some or no digits may be added to the back of `num`. No digits may be inserted in the middle of `num`. Remember, exactly `numDigits` digits have been adjoined to num.
  Or, in code:
  `("" + answer).length() == ("" + num).length() + numDigits`
- `answer` is divisible by all values in `divisors`.

For example:

`adjoinDigits(523, 3, new int[] {7, 8, 9}) returns 155232`. The three digits 1, 5, and 2 have been adjoined to 523. The 1 and the 5 have been adjoined to the front and the 2 has been adjoined to the back. `155232` is divisible by 7, 8 and 9. And `155232` is the smallest such number.

The following code shows the results of the `adjoinDigits` method.

The following code	Returns
`AdjoiningDigits.adjoinDigits(523, 3, new int[] {7, 8, 9} );`	`155232`
`AdjoiningDigits.adjoinDigits(77, 1, new int[] {13, 29} );`	`377`
`AdjoiningDigits.adjoinDigits(32, 2, new int[] {11, 13, 23} );`	`3289`
`AdjoiningDigits.adjoinDigits(50, 2, new int[] {2, 5} );`	`1050`

- I do not know if it is possible that no such number exist for the `adjoinDigits` method, but you may assume a number will exist for every call to the `adjoinDigits` method.