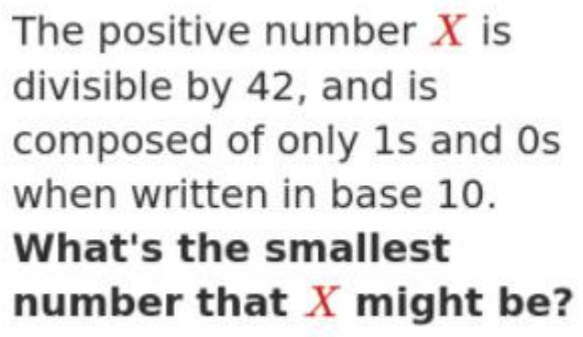
Digit Guardian

The motivation for this problem comes from the similar puzzle:


The positive number X is divisible by 42, and is composed of only 1s and 0s when written in base 10.
What's the smallest number that X might be?

This problem is concerned with base 10 integers. We say that that base 10 numbers are constructed with the (standard) decimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

In this problem you will complete three methods in the `DigitGuardian` class. The `DigitGuardian` class has an `ArrayList<Integer> digits` as an instance variable. The instance variable is initialized by the constructor. The precondition is all values in `digits` are single decimal digit.

The first method to complete is: `boolean isAllowable(int num). isAllowable` returns true if both:

- Every decimal digit in `digits` appears at least once in `num`
  Leading zeros cannot satisfy this condition.
- And `num` contains only decimal digits in `digits`.

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

The following code	Returns

`List<Integer> digs = new ArrayList<Integer>();`
`digs.add(new Integer(0));`
`digs.add(new Integer(2));`
`digs.add(new Integer(5));`
`digs.add(new Integer(8));`
`DigitGuardian dg = new DigitGuardian(digs);`	

`dg.isAllowable(5082);`	`true`
`dg.isAllowable(50852);`	`true`
`dg.isAllowable(582);     //` does not contain a 0	`false`
`dg.isAllowable(12508);   //` contains a 1, not in `digits`	`false`

Digit Guardian

The second method to complete is: `getDigits(int num)`. `getDigits` returns a `List<Integer>` containing every decimal digit in `num` in any order.
Please note:

- Order of the digits in the List is not important.
- Duplicate decimal digits in `num` are not repeated in the returned List.
- The contents of the instance variable `digits` are NOT relevant in this problem.

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

The following code	Returns

`List<Integer> digs = new ArrayList<Integer>();`
`/*  code not shown  */`
`DigitGuardian dg = new DigitGuardian(digs);`
 	
`List<Integer> ans = dg.getDigits(1575);`	

`ans.size();`	3
`ans.contains(new Integer(1));`	true
`ans.contains(new Integer(5));`	true
`ans.contains(new Integer(7));`	true
`ans.contains(new Integer(0));`	false

The third and final method to complete is: `int getMinLCM(int divisor)`. `getMinLCM` returns smallest number, `ans`, that is allowable `(isAllowable(ans) == true)` and divisible by `divisor` `(ans % divisor == 0)`.

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

The following code	Returns

`List<Integer> digs = new ArrayList<Integer>();`
`digs.add(new Integer(0));`
`digs.add(new Integer(2));`
`digs.add(new Integer(5));`
`digs.add(new Integer(8));`
`DigitGuardian dg = new DigitGuardian(digs);`	

`dg.getMinLCM(2);     //  isAllowable(2058) == true`
`                     //  && 2058 % 2 == 0`
	2058

`dg.getMinLCM(97);    //  isAllowable(5820) == true`
`                     //  && 5820 % 97 == 0`
	5820

`dg.getMinLCM(117);   //  isAllowable(2025855) == true`
`                     //  && 2025855 % 117 == 0`
	2025855