DecryptMe

In this problem you will implement four methods in the `DecryptMe` class. The four methods are `countLetter(), groupCounter(), getBestMessage(), numDecryption().`

This problem was asked by Facebook.
Given the mapping a = 1, b = 2, ... z = 26, and an encoded message (containing only the number 1 to 26 that map to the lower case letters) count the maximum number of times a specific letter may appear.

The `countLetter(String mess, int let )` method returns the maximum number of times a specific letter `(let)` may appear in an encoded message (`mess`).

For example, the message `"111"` contains at most 3 a's, and at most 1 k.

Note: 0 is not a valid value. Therefore, you may not count a 1 or 2 if it precedes a 0. The remaining numbers (3, 4, 5, … 9) will never precede a 0. This implies the message:

- `"110120"` does **not** contain any k's (11) or l's (12) and contains exactly 2 a's (1), 1 j (10), and 1 t (20)
- `"1020"` does **not** contain any a's (1) and does **not** contain any b's (2), and contains exactly 1 j (11) and 1 t (20).

The following code shows the results of the `countLetter(mess, n)` method.

The following code	Returns
`DecryptMe.countLetter("111", 1);`	3
`DecryptMe.countLetter("111", 11);`	1
`DecryptMe.countLetter("110120", 1);`	2
`DecryptMe.countLetter("110120", 10);`	1
`DecryptMe.countLetter("110120", 11);`	0
`DecryptMe.countLetter("110120", 12);`	0
`DecryptMe.countLetter("110120", 20);`	1
`DecryptMe.countLetter("1020", 1);`	0
`DecryptMe.countLetter("1020", 2);`	0
`DecryptMe.countLetter("1020", 10);`	1
`DecryptMe.countLetter("1020", 20);`	1

DecryptMe

The `groupCounter(String mess, int[] lets)` method counts the maximum number of times a group of letter appears. That is, the max sum of the number of times each letter in the `int[] lets` could be in `mess`. {123 – may only count the 12 letter or 23 letter, NOT both} You may assume the letters in the parameter `lets` are in ascending order (that is, increasing)

Remember: 0 is not a valid value. Therefore, you may not count a 1 or 2 if it precedes a 0. The remaining numbers (3, 4, 5, … 9) will never precede a 0.

The following code shows the results of the `groupCounter(String mess, int[] lets)` method.

The following code	Returns
`DecryptMe.groupCounter("111", new int[] {1, 2, 3, 23});`	3
`DecryptMe.groupCounter("123", new int[] {2, 12, 23});`	1
`DecryptMe.groupCounter("2317", new int[] {2, 3, 17});`	3
`DecryptMe.groupCounter("12010715", new int[] {1, 2, 7, 15});`	3

The `getBestMessage(String[] messages, int[] lets)` returns the message or messages in `messages` with the maximum sum of possible occurrences of the letters contained in `lets`. Once again, you assume the values in `lets` are in ascending order. All messages with the maximum sum must be returned in List of `String`s.

The following code shows the results of the `getBestMessage(messages, lets)` method.

The following code	Returns

`String[] messages = { "12345", "1111", "12233", "223435"};`
 	
`ArrayList<String> ans = DecryptMe.getBestMessage(messages,`
`new int[] {1, 2, 3, 23});`	

`ans.size();`	1
`ans.get(0);`	"12233"

Turn to next page for another example of the `getBestMessage(messages, lets)` method.

DecryptMe

The following code shows the results of the `getBestMessage(messages, lets)` method.

The following code	Returns

`String[] mess1 = { "1523423735", "221323151517", "172323513", "7223423315"};`
 	
`ArrayList<String> ans = DecryptMe.getBestMessage( mess1, new int[] {3, 7, 15, 23});`	

`ans.size();`	3
`ans.contains("1523423735"));`	true
`ans.contains("221323151517"));`	true
`ans.contains("7223423315"));`	true

The `numDecryption(String mess)` method returns the number of ways the parameter `mess` can be decoded.

For example:

- `"111"` returns 3, since it could be decoded as `"aaa"`, `"ka"`, and `"ak"`.
- `"1310"` return 2, since it could be decoded as `"mj"`, and `"acj"`.

You can assume that the messages are decodable. That is, "001". is not an allowable value for the parameter `mess`.

The following code shows the results of the `numDecryption(mess)` method.

The following code	Returns
`DecryptMe.numDecryption("111");`	3
`DecryptMe.numDecryption("1310");`	2