Problem Statement

You are given a int[] **marks** containing the grades you have received so far in a class. Each grade is between 0 and 10, inclusive. Assuming that you will receive a 10 on all future assignments, determine the minimum number of future assignments that are needed for you to receive a final grade of 10. You will receive a final grade of 10 if your average grade is 9.5 or higher.

Definition

Class:	AimToTen
Method:	need
Parameters:	int[]
Returns:	int
Method signature:	int need(int[] marks)

(be sure your method is public)

Constraints

- **marks** has between 1 and 50 elements, inclusive.
- Each element of **marks** is between 0 and 10, inclusive.

Examples

0)

```
{9, 10, 10, 9}
```

```
Returns: 0
```

Your average is already 9.5, so no future assignments are needed.

1)

```
{8, 9}
```

```
Returns: 4
```

In this case you need 4 more assignments. With each completed assignment, your average could increase to 9, 9.25, 9.4 and 9.5, respectively.

2)

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Returns: 950
```

3)

```
{10, 10, 10, 10}
Returns: 0
```

Problem Statement

A sequence of characters is called a *fence* if it consists of alternating '`|`' and '`-`' characters, like "`|-|-|-|`" or "`-|-|`" (quotes for clarity only). Notice that "`|-||-|`" or "`--`" are not fences, because each contains two equal characters adjacent to each other. Given a string **s**, find the longest consecutive substring of it that is a fence, and return its length.

Definition

Class:	FunnyFence
Method:	getLength
Parameters:	String
Returns:	int
Method signature:	int getLength(String s)

(be sure your method is public)

Constraints

- **s** will contain between 1 and 50 characters, inclusive.
- Each character of **s** will be either '`|`' or '`-`'.

Examples

0)

```
"|-|-|"
Returns: 5
```

The entire string is a fence.

1)

```
"-|-|-|-"
Returns: 7
```

Still a fence.

2)

```
"||||||"
Returns: 1
```

A fence can be just 1 character long, so every 1 character substring here is a fence.

3)

```
"|-||-|-"
Returns: 4
```

The last 4 characters form the longest consecutive substring that is a fence.

4)

```
"|-|---|-|---|-|"
Returns: 5
```

"-|-|-" right in the middle gives the longest fence.

5)

```
"|||-||--|--|---|-||-|-|-|--||---||-||-||-|--||"
Returns: 8
```

Problem Statement

We call two numbers friendly if they have the same digits, ignoring order or repetition. For example 122213 and 312 are friendly while 145 and 2544411 are not. A sequence is friendly if it contains at least two numbers, and all possible pairs of numbers within it are friendly. Two contiguous subsequences are different if they have a different start index, end index or both.

If we are given the sequence 112, 12, 21, 354, 534345, 345, 2221 then the friendly contiguous subsequences are: {112, 12}, {112, 12, 21}, {12, 21}, {354, 534345}, {354, 534345, 345} and {534345, 345}. {112, 12, 21, 354} is not a friendly contiguous subsequence because 112 and 354 are not friendly numbers and {112, 12, 21, 2221} is not a friendly contignuous subsequence because the elements of the sequence aren't in consecutive positions in the original sequence.

Given a int[] **array**, you must return number of different friendly contignuous subsequences of **array**.

Definition

Class:	FriendlySequences
Method:	count
Parameters:	int[]
Returns:	int
Method signature:	int count(int[] array)

(be sure your method is public)

Constraints

- **array** will have between 0 and 50 elements, inclusive.
- Each element of **array** will be between 0 and 2000000000, inclusive.

Examples

0)

```
{112, 12, 21, 354, 534345, 345, 2221}
Returns: 6
```

The example in the problem.

1)

```
{10, 1100, 10101, 111, 1111, 11111, 11, 1, 111}
Returns: 18
```

2)

```
{0, 0, 0, 0}
Returns: 6
```

We have a total of 6 possible different pairs of start and end indices for friendly subsequences.

3)

```
{123456890, 213456890, 198654320}
Returns: 3
```

4)

```
{9}
Returns: 0
```