

# Test 1 Supplement

October 12, 2009

## Credit Crunch Revisited

*This problem can be used as a supplement for the work you did on Test 1.*

Credit card numbers can be validated using what's called the *Luhn's checksum algorithm*. The algorithm basically works as follows, where the left-most digit has index 0 (as it does for strings) and the algorithm works from left to right examining each digit of the credit card.

Accumulate a sum based on adding a value obtained from each digit of the credit card as follows, where each  $k^{th}$  digit is examined starting with  $k = 0$ .

- If  $k$  is even, add the  $k^{th}$  digit to the sum.
- If  $k$  is odd, multiply the  $k^{th}$  digit by two. If the result is  $\geq 10$ , subtract 9. Add this computed value to the sum.

If the resulting sum is divisible by 10, the credit card number passes the checksum test, otherwise the card number isn't valid. Credit card numbers are often 16-digits long, but they can be of any length.

Here's an example for the card number 2543-2109-8765-4321 showing that it is a valid card number.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
credit card number	2	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1
value added	2	10-9	4	6	2	2	0	18-9	8	14-9	6	10-9	4	6	2	2
total	2	3	7	13	15	17	17	26	34	39	45	46	50	56	58	60

Since this is valid, the call `isValid({2,5,4,3,2,1,0,9,8,7,6,5,4,3,2,1})` evaluates to "YES". However, the call `isValid({1,2,3,4})` evaluates to "NO" since the computed sum is  $1 + 4 + 3 + 8 = 16$  which isn't divisible by 10.

Write the method `isValid` whose header is given below and submit under `test1-supp`.

```
public class Luhn {
    /**
     * Returns "YES" if credit-card number's checksum is valid
     * @param ccard contains only values in [0-9]
     * @return "YES" if ccard's checksum is valid, otherwise returns "NO"
     */
    public String isValid(int[] ccard)
    {
        // TODO: fill in code here
    }
}
```

# APT: Luhn Algorithm

## Problem Statement

The *Luhn Checksum* helps validate credit card numbers (Mastercard, Visa, American Express, etc.), Canadian Social Insurance Numbers, and several other numbers. You must research how the Luhn algorithm (also called the mod 10 algorithm) calculates validity and write a method to determine if a sequence of digits stored in an array is valid according to this method.

The parameter `number` is a sequence of digits, the element at index zero represents the left-most digit, the right-most digit is the last element in the array.

The method `isValid` should return the String "YES" if the array `number` represents a valid sequence according to the Luhn algorithm and "NO" if the sequence is not valid.

### Class

```
public class Luhn {  
    public String isValid(int[] number) {  
        // fill in code here  
    }  
}
```

## Notes and Constraints

- The array `number` represents a sequence of digits to be checked for validity. The element at index zero of the array represents the left-most digit in the sequence.
- Parameter `number` will have at least one element and no more than 50 elements.
- Each value in `number` will be between 0 and 9 inclusive.

## Examples

1. `number = [1,2,1,4]`  
Returns: "YES"

The Luhn algorithm generates a sum from right-to-left of  $4 + 1*2 + 2 + 1*2 = 10$  which is divisible by 10 and thus a valid number/sequence.

2. `number = [1,7,2,3,6]`  
Returns: "YES"

The Luhn algorithm generates (from right-to-left)  $6 + 6 + 2 + 5 + 1 = 20$  which is divisible by 10.

3. `number = [7,8,3]`  
Returns: "NO"

The Luhn algorithm generates (from right-to-left)  $3 + 7 + 7 = 17$  which is not divisible by 10.



This work is copyright © Owen Astrachan and is licensed under a [Creative Commons Attribution-Share Alike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/).