KATA: Valid credit card

Create a function that takes a number as an argument and returns true if the number is a valid credit card number, false otherwise.

Credit card numbers must be between 14-19 digits in length, and pass the Luhn test, described below:

1. Remove the last digit (this is the "check digit").
2. Reverse the number.
3. Double the value of each digit in odd-numbered positions. If the doubled value has more than 1 digit, add the digits together (e.g. 8 x 2 = 16 ➞ 1 + 6 = 7).
4. Add all digits.
5. Subtract the last digit of the sum (from step 4) from 10. The result should be equal to the check digit from step 1.

**Examples**

ValidateCard(1234567890123456) ➞ false

// Step 1: check digit = 6, num = 123456789012345

// Step 2: num reversed = 543210987654321

// Step 3: digit array after selective doubling: [1, 4, 6, 2, 2, 0, 9, 8, 5, 6, 1, 4, 6, 2, 2]

// Step 4: sum = 58

// Step 5: 10 - 8 = 2 (not equal to 6) ➞ false

ValidateCard(1234567890123452) ➞ true

// Same as above, but check digit checks out.

**Resources**

Luhn Wikipedia (this algorithm is what the kata is based on):

<https://en.wikipedia.org/wiki/Luhn_algorithm>

Credit card number generator:

<https://www.freeformatter.com/credit-card-number-generator-validator.html>

**Tests**

Assert.Equals(false, Challenge.ValidateCard(79927398714));

Console.WriteLine("Passes Luhn test, but too short.");

Assert.Equals(false, Challenge.ValidateCard(79927398713));

Assert.Equals(true, Challenge.ValidateCard(709092739800713));

Assert.Equals(false, Challenge.ValidateCard(1234567890123456));

Assert.Equals(true, Challenge.ValidateCard(12345678901237));

Assert.Equals(true, Challenge.ValidateCard(5496683867445267));

Assert.Equals(false, Challenge.ValidateCard(4508793361140566));

Assert.Equals(true, Challenge.ValidateCard(376785877526048));

Assert.Equals(false, Challenge.ValidateCard(36717601781975));