Checking Credit Cards

Introduction

Your task is to develop a REST API that validates different kinds of credit cards.

If you are applying for a role with a specific language specialisation, you should use that language. Otherwise you should complete the test in one of our standard backend-languages (Java, GoLang or JavaScript). You are welcome to use TypeScript as well.

You should write clean, concise code adhering to normal best practices for your chosen language with appropriate test coverage.

You should write readme documentation that tells us at a minimum how to run your project and how to access any endpoints you have created.

Specific design decisions are up to you. We have outlined a minimum set of requirements but it is up to you to consider how you design and implement a solution. You should consider ways you can improve the solution (e.g. by improving test cases or finding other optimisations) and either implement them (if you have time) or by telling us about these improvements.

Problem

Before submitting a credit card to a payment gateway it's important that we run some sanity checks on the number

A common check that is performed upfront is to validate the card type based on the starting digits and length of card number. The main patterns that we care about are as follows:

Card Type	Begins With	Number Length
AMEX	34 or 37	15
Discover	6011	16
MasterCard	51-55	16
Visa	4	13 or 16

All of these card types also generate numbers such that they can be validated by the **Luhn Algorithm**, so that's the second check systems usually try.

The steps for validating using the Luhn algorithm are:

1. Starting with the next to last digit and continuing with every other digit going back to the beginning of the card, double the digit.

2. Sum all doubled and untouched digits in the number. For digits greater than 9 you will need to split them and sum them independently (i.e. "10", 1 + 0).

3. If that total is a multiple of 10, the number is valid.

For example, let's assume the given card number is 4408 0412 3456 7893:

1: 8 4 0 8 0 4 2 2 6 4 10 6 14 8 18 3
2: 8+4+0+8+0+4+2+2+6+4+1+0+6+1+4+8+1+8+3 = 70
3: 70 % 10 == 0

Thus this card is valid.

Let's try one more, 4417 1234 5678 9112:
1: 8 4 2 7 2 2 6 4 10 6 14 8 18 1 2 2
2: 8+4+2+7+2+2+6+4+1+0+6+1+4+8+1+8+1+2+2 = 69
3: 69 % 10 != 0

Therefore this card is not valid.

Input and Output

Given the following credit cards:
4111111111111111
4111111111111
4012888888881881
378282246310005
6011111111111117
5105105105105100
5105 1051 0510 5106
9111111111111111

I would expect the following output:

VISA: 4111111111111111 (valid)
VISA: 4111111111111 (invalid)
VISA: 4012888888881881 (valid)

AMEX: 378282246310005 (valid)
Discover: 6011111111111117 (valid)
MasterCard: 5105105105105100 (valid)
MasterCard: 5105 1051 0510 5106 (invalid)
Unknown: 9111111111111111 (invalid)

Submission

You must use version control and you will need to send through both a .zip file containing all of your code (including its history) and a link to a publicly accessible repository (e.g. github) where we can we view your code.