Introduction

Hello fellow Code Enthusiast! 🎉

Thanks for your interest in Wunder! The next stage of our process is to ask you to complete a short exercise in any language. Please follow the instructions below and return the results by sending a link to your repository via email.

Your take home challenge consists of two parts, which you will find in the following pages.

- *Scooter parsing I*
- *Scooter parsing II*

You are allowed to use any Language, Framework, and Library to solve this challenge. If you have any questions regarding the take home challenge, please don't hesitate to contact us for further clarification! We are always happy to help.

Checklist

In order to answer some questions upfront which might come up, we included a small checklist with tasks you might consider when solving this challenge:

✅ Add unit tests

✅ Upload solution to a Repository (e. g. GitHub, BitBucket, GitLab)

✅ Send us the link to your new Repository

Scooter Parsing (PART I)

Your Assignment is to write a function which returns the device informations of a payload stream that a Wunder Kick-Scooter sent us.

Payload Structure

Here you see a chart that describes the Payload Structure of a typical Wunder Kick-Scooter packet. The chart precisely details the positions of the respective parameters.

Position	Parameter	Length (Byte)	Format
Begin of Packet	Command	1	+
0	Type	2-3	IN / OUT
1	Instruction	<= 20	DeviceInfo / PositionUpdate / etc.
3	Unique ID	15	IMEI
4	BatteryLevel	1 - 3	0 - 100
4	Odometer	<= 6	0 - 999999
6	Time	19	ISO date 'YYYY-MM-DDTHH:MM:SS'
7	Count Number	4	0000 - FFFF
End of Packet	Tail Character	1	$\n

Example Payload Stream

```
'
+IN,DeviceInfo,860861040012977,86,5600,2021-01-14T15:05:10,0035$
AABBAA
+IN,DeviceInfo,860861040012977,34,5612,2021-01-14T18:30:10,0036$
CCDDEE
+IN,DeviceInfo,860861040012977,3,5623,2021-01-14T23:59:10,0037$
FFGGHH
'
```

Assignment

Your Assignment is to write a function, which parses a given scooter payload stream. The function should return an ARRAY consisting of OBJECTS which contain `imei`, `batteryLevel`, `odometer`, and `time`.

Example:

```
// 1. input
const payloadStream = `
+IN,DeviceInfo,860861040012977,86,5600,2021-01-14T15:05:10,0035$
AABBAA
+IN,DeviceInfo,860861040012977,34,5612,2021-01-14T18:30:10,0036$
CCDDEE
+IN,DeviceInfo,860861040012977,3,5623,2021-01-14T23:59:10,0037$
FFGGHH
`

// 2. calling
getDeviceInformations(payloadStream)

// 3. returns
[
  { imei: '860861040012977', batteryLevel: '86 %', odometer: '5600 km',
time: 2021-01-14T14:05:10.000Z },
  { imei: '860861040012977', batteryLevel: '34 %', odometer: '5612 km',
time: 2021-01-14T17:30:10.000Z },
  { imei: '860861040012977', batteryLevel: '3 %', odometer: '5623 km',
time: 2021-01-14T22:59:10.000Z },
]
```

Task:

```
/*
 * The function accepts a STRING `payloadStream` as parameter.
 *
 *
 * The function is expected to RETURN an ARRAY in the following format:
 *
 *  [{ imei: STRING, batteryLevel: STRING, odometer: STRING, time: DATE },
... ]
 *
 */
function getDeviceInformations(payloadStream) {

  // your code here...

}
```

Scooter Parsing (PART II)

Extend your code form Part I, so it also parses multiple different packages, which have a dynamic length.

Payload Structure

Here you see two charts. One describes the DeviceInfo packet and the other the Error packet. Those charts precisely detail the positions of the respective parameters.

DeviceInfo

Position	**Parameter**	**Length (Byte)**	**Format**
Begin of Packet	Command	1	+
0	Type	2-3	IN / OUT
1	Instruction	<= 20	DeviceInfo / PositionUpdate / etc.
2	Unique ID	15	IMEI
3	BatteryLevel	1 - 3	0 - 100
4	Odometer	<= 6	0 - 999999
5	Time	19	ISO date 'YYYY-MM-DDTHH:MM:SS'
6	Count Number	4	0000 - FFFF
End of Packet	Tail Character	1	$\n

Error

Position	**Parameter**	**Length (Byte)**	**Format**
Begin of Packet	Command	1	+
0	Type	2-3	IN / OUT
1	Instruction	5	Error
2	Unique ID	15	IMEI
3	`Total Errors`	1	1 - 9, describes how many Error Tuples (`Error Code`* + `Error Description`*) within that packet exist.

Position	Parameter	Length (Byte)	Format
n	`Error Code`	1 - 2	0 - 99
n + 1	`Error Description`	<= 20	NoBattery / ECUFailure / etc.
n + 1 + (`Total Errors` * 2)	Time	19	ISO date 'YYYY-MM-DDTHH:MM:SS'
n + 2 + (`Total Errors` * 2)	Count Number	4	0000 - FFFF
End of Packet	Tail Character	1	$\n

Example Payload Stream

```
'
+IN,DeviceInfo,860861040012977,86,5600,2021-01-14T15:05:10,0035$
AABBAA
+IN,Error,860861040012977,2,5,NoBattery,7,ECUFailure,2021-01-
14T15:06:18,0036$
+IN,Error,860861040012977,1,7,ECUFailure,2021-01-14T15:09:18,0037$
+IN,DeviceInfo,860861040012977,34,5612,2021-01-14T18:30:10,0038$
CCDDEE
+IN,Error,860861040012977,4,5,NoBattery,7,ECUFailure,8,Reboot,10,IotError,
2021-01-14T19:05:10,0039$
+IN,DeviceInfo,860861040012977,3,5623,2021-01-14T23:59:10,0040$
FFGGHH
'
```

Assignment

Your Assignment is to extend your function from PART I, so it also parses various packages, which have a dynamic length.

Example:

```
// 1. input
const payloadStream = `
+IN,DeviceInfo,860861040012977,86,5600,2021-01-14T15:05:10,0035$
AABBAA
+IN,Error,860861040012977,2,5,NoBattery,7,ECUFailure,2021-01-
14T15:06:18,0036$
+IN,Error,860861040012977,1,7,ECUFailure,2021-01-14T15:09:18,0037$
+IN,DeviceInfo,860861040012977,34,5612,2021-01-14T18:30:10,0038$
```

```
CCDDEE
+IN,Error,860861040012977,4,5,NoBattery,7,ECUFailure,8,Reboot,10,IotError,
2021-01-14T19:05:10,0039$
+IN,DeviceInfo,860861040012977,3,5623,2021-01-14T23:59:10,0040$
FFGGHH
`

// 2. calling
getDeviceInformations(payloadStream)

// 3. returns
[
  { imei: '860861040012977', batteryLevel: '86 %', odometer: '5600 km',
time: 2021-01-14T15:05:10.000Z },
  { imei: '860861040012977', NoBattery: 2, ECUFailure: 7, time: 2021-01-
14T15:06:18.000Z },
  { imei: '860861040012977', ECUFailure: 7, time: 2021-01-14T15:09:18.000Z
},
  { imei: '860861040012977', batteryLevel: '34 %', odometer: '5612 km',
time: 2021-01-14T18:30:10.000Z },
  { imei: '860861040012977', NoBattery: 5, ECUFailure: 7, Reboot: 8,
IotError: 10, time: 2021-01-14T19:05:10.000Z },
  { imei: '860861040012977', batteryLevel: '3 %', odometer: '5623 km',
time: 2021-01-14T23:59:10.000Z },
]
```

Task:

```
/*
 * The function accepts a STRING `payloadStream` as parameter.
 *
 * The function is expected to RETURN an ARRAY of Objects. Please refer to
Example for additional informations.
 */
function getDeviceInformations(payloadStream) {

  // Please copy your code from PART I.

}
```

Thank you for your time, good luck! 🍀