Software Engineering Test

Minimum Requirement

1. Your beloved IDE (VSCode, Atom, Notepad++, etc)
2. Your programming language (PHP, Go, Python, Javascript, etc)
3. Knowledge basic about Logical Thinking
4. Knowledge basic about REST API

Challenge 1

1. Write a program that can receive number (n) from 1 to 100.
2. Iterate from `1` to `n` (from point 1) with the following conditions:
    - If the iteration number is a multiples of 3 append `Mc`
    - If the iteration number is a multiples of 5 append `Easy`
    - If the iteration number is a multiples of 3 and 5 append `McEasy`
    - otherwise append the iteration number itself
3. The output should be in an array type.

Example:

Input: `n = 17`

Output: `[1,2,"Mc",4,"Easy","Mc",7,8,"Mc","Easy",11,"Mc",13,14,"McEasy",16,17]`

NOTE: If you are using static typing languange (like: Go, Java, C, C++) the output can be array of string for example: `["1","2","Mc","4"]`

\

Challenge 2

1. Create a function that receive an array type.
2. Based on Challenge-1 output filter the `"Mc"` and `"Easy"` word only.
3. The output should be in an array type.

Example:

Input: `[1,2,"Mc",4,"Easy","Mc",7,8,"Mc","Easy",11,"Mc",13,14,"McEasy",16,17]`

Output: `["Mc","Easy","Mc","Mc","Easy","Mc"]`

NOTE: If you are using static typing languange (like: Go, Java, C, C++) the input can be array of string for example: `["1","2","Mc","4"]`

\

Challenge 3

1. From the Challange 2 output, validate how much `"Mc"`, `"Easy"` pair in the array.
2. The pair should be ordered but it can jumped too, that means `["Easy","Mc"]` has `0` pair but `["Mc","Mc","Easy","Easy"]` has `2` pairs.
3. The output is an integer that represent total match pairs.

Example 1:

Input: `["Mc","Easy"]`

Output: `1`

Example 2:

Input: `["Easy","Mc"]`

Output: `0`

Example 3:

Input: `["Easy","Mc","Easy"]`

Output: `1`

Example 3:

Input: `["Mc","Mc","Easy","Easy"]`

Output: `2`

Example 4:

Input: `["Mc","Mc","Easy","Mc","Easy"]`

Output: `2`

Example 5:

Input: `["Mc","Easy","Mc","Mc","Easy","Mc","Easy","Mc","Mc","Easy","Mc","Easy","Easy"]`

Output: `6`

\

Challenge 4

Expose Challenge-3 into an REST API with the following condition:

Request:

1. Method: `POST`
2. Endpoint: `/api/mceasy`
3. Request header: `Content-Type: application/json`
4. JSON Request:

```
{
    "arr":
["Mc","Easy","Mc","Mc","Easy","Mc","Easy","Mc","Mc","Easy","Mc","Easy","Easy"]
}
```

Response:

1. Response header: `Content-Type: application/json`
2. JSON Response:

```
{
    "valid_pair": 6
}
```

Expected behaviour:

Windows:

```
$ curl -X POST -H "Content-Type: application/json" -d "{\"arr\":
[\"Mc\",\"Easy\",\"Mc\",\"Mc\",\"Easy\",\"Mc\",\"Easy\",\"Mc\",\"Mc\",\"Easy\",\"Mc\",\'
 "http://localhost:3000/api/mceasy"
{"valid_pair": 6}
```

Unix/Linux:

```
$ curl -X POST -H 'Content-Type: application/json' -d '{"arr":
["Mc","Easy","Mc","Mc","Easy","Mc","Easy","Mc","Mc","Easy","Mc","Easy","Easy"]}'
http://localhost:3000/api/mceasy
{"valid_pair": 6}
```

Challenge 5: Dockerization (Optional)

Create a `Dockerfile` from the Challenge 4 so that your application can shipped everywhere and across platform

Expected behaviour:

```
$ docker build . -t mceasy.co.id/software-engineer-test:v1.0
$ docker run -d -p 3000:3000 -e "PORT=3000" mceasy.co.id/software-engineer-test:v1.0
```