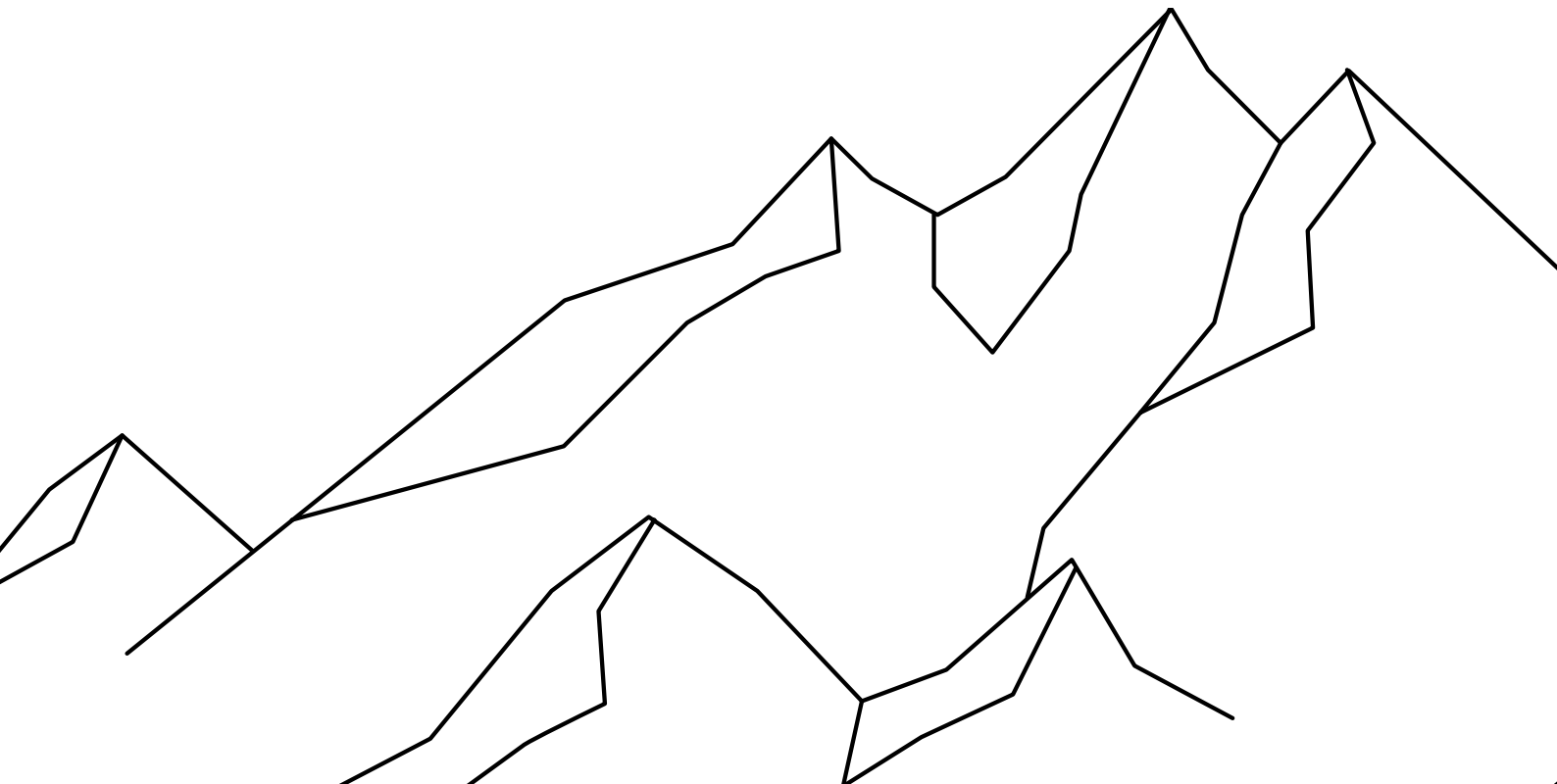
Alpian Technologies

MOBILE SOFTWARE ENGINEER TEST



Software Engineer Technical Test

Congratulations on getting to this stage and thank you for taking the time to solve the following exercises. Please remember to commit your code frequently using a version control system of your choice. Once you successfully complete this tasks please share your repository with us.

In your repository you should include:

- A text file with your thought process (for every question that require a description)
- The source code to solve the exercise
- A readme file with all the instructions that you think will be useful for us to test your solution.

Best of luck.

NOTE: Keep in mind that you should keep your repository private. You can use https://github.com to create an account with free private repos. Once you are done you can share your repo with ***Tiamaroth*** **,** ***eugenekup*** **,** ***LalinP*** and ***lao1g12*** on GitHub. Please remember to create a sub-folder and clearly mention your Name & Surname to allow a quick identification.

Exercise 1

Some programming languages do not have loops; instead they make use of recursion. Therefore, it is good to be fluent in translating a function with loops into a recursive one.
Here is an iterative function collatz written in pseudocode (this function is the well-known Collatz conjecture, https://en.wikipedia.org/wiki/Collatz_conjecture):

```
collatz(n)
while n > 1
    show n
    if n odd
        set n to 3n + 1
    else
        set n to n / 2
```

I. When written recursively, the function body can be reduced to a single *return* statement.

a. Express this function as a recursive one in any language of your choice (Java, Go, Kotlin etc).
b. Rewrite your answer so that it is tail-recursive in any language of your choice (Java, Go, Kotlin ,etc).

Exercise 2

In computer graphics an operation called the *dotproduct* is used to manipulate vectors. The dot product of (a1, a2,⋯, an) and (b1, b2,⋯, bn) is a1b1+a2b2+⋯+anbn; using the zip, map and reduce operations, write a function *dotProduct* that computes the dot product of two vectors in any language that implements zip, map and reduce functions on your choice.

Exercise 3

You're tasked to implement the new customer-mapper-service in a language of your choice among Java, Go or Kotlin.

This service stores entities defined as:

- customerId -> int
- externalId -> string
- createdAt -> date (you are free to use the most convenient date type of the language you chose)

It will expose two endpoints with a POST and GET operations.

- POST endpoint will take as parameter the customerId and the createdAt, store it in a cache / in-memory database (up to you!) pairing it with an externalId. The externalId can be generated it in the service itself. The date is expected to be valid with format *yyyy-mm-dd* and can't be in the future.

- GET endpoint will return the externalId of a given customerId

Endpoints can be either Rest or gRPC, your choice.