HOMA GAMES
FULL STACK DEVELOPER
CASE STUDY

Task WordFinder

- Let **L** be a list of words made of letters from the classical Latin alphabet **A: string[] = ['a', 'b', 'c', ..., 'z']**.
- Let **s** be a word of at most twelve letters from the alphabet **A**.
- Consider the function ***longestWord*** that, given a word **s**, returns the longest word from **L** that can be obtained using a subset of the letters from **s**. The goal of this exercise is to write a class called ***WordFinder*** that efficiently implements this function.
- For example, given **L: string[] = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune']** and **s: string = 'ajsxuytcnhre'**, the function ***longestWord*** returns **saturn**. Note that returning **uranus** would be wrong as **s** only contains one letter **u**.

Example template in TypeScript

```
const dictionaryList = /* TODO */;
const longestWord = (/* TODO */): string => { /* TODO */ };

class TaskWordFinder {
    public longest: string;

    public longestWordFinder(/* TODO */) {
        // TODO
        this.longest = longestWord(/* TODO */);
        // TODO
    }
}
```

- Please:
  - Create a constant dictionaryList as the dictionary collection. This collection should contain a list **L** of words, with a single word per line.
  - Write a function ***longestWord*** that implements the following:
    - Taking as argument a word **s** of at most twelve letters from **A**, and returning the longest word from **L** that can be built using the letters from **s**. If several words with the same longest length can be built, the method should simply return one of them.
  - Write a class ***TaskWordFinder*** whose ***longestWordFinder*** method takes two arguments:
    - A local file name giving the dictionary of words to choose from. As an example, you can use the file: https://goo.gl/aoEr9Q
    - A word of at most twelve letters from the alphabet **A**.
  - The ***longestWordFinder*** method should save the longest word from the dictionary using the letters of the input word.
  - Do not write comments.
  - Provide an **efficient** solution in TypeScript.

- In order to implement an optimal solution, it is important to choose the most suitable data structures and algorithms. Feel free to use any standard Typescript or Node SDK for your project.

- Example input: google-10000-english.txt optonoceari
  Example output: cooperation

- Please make sure that the code compiles and runs "as is" using the standard command line programs (e.g. node) and provide the commands required to compile and run your code

Task SQL

- Let us consider the following two relational tables:
  - The table **PLAYERS**, which gives information about football players: team, homeland and market value in millions.

player_id	team	homeland	market_value
...	...	...	...
34546	FCB	argentina	12.2
34547	FCB	spain	8.6
34548	PSG	brazil	9.8
...	...	...	...

  - The **GOALS** table, which gives information about scored goals: ID of the striker, timestamp of the goal, and its beauty from 0.0 to 1.0.

goal_id	player_id	timestamp	beauty
...	...	...	...
167	34546	2016-11-23 00:10:03	0.89
158	34548	2016-11-23 02:56:32	0.27
...	...	...	...

A. Please write a query that returns:

1. The goals whose beauty is above 0.9 or below 0.1.
2. The players of 'FCB' who have scored at least 1 goal.
3. The players of 'FCB' whose market valuation is lower than 8.0 and who scored at least 10 goals.
4. The number of goals scored for each team and homeland on Nov. 22nd 2016.
5. The average goal beauty over all the players of the 'PSG' team.

B. He wants to write a query that returns the couples: (player_id, goal_id) for every player. However, the following query:

```
SELECT A.player_id, B.goal_id
FROM PLAYERS AS A, GOALS AS B
WHERE A.player_id = B.player.id
```

only returns the couples (player_id, goal_id) for every scored goal. Why is that? How would you rewrite the query to return the desired result? Please explain.

C. How would you rewrite the following query without a subquery?

```
SELECT COUNT(*), team
FROM PLAYERS AS A
WHERE A.player_id NOT IN (SELECT B.player_id FROM GOALS AS B)
GROUP BY team
```