Kaffa Mobile - Pre-qualification test (v1.10)

Objective

Complete **as many** exercises as you can.

Important:

- provide a **README.md** with instructions on how to build and run your code;
- add **screenshots / gifs / videos** to **README.md** showing your applications running;
- if you don't know how to complete an exercise, leave it blank. But the more exercises you solve correctly, the better;
- you will be compared to other candidates, so make sure to show your skills. Feel free to do more than asked;
- complete the tasks yourself without help and **without copying and pasting** code from the internet. Don't submit code you don't understand, you'll be asked in the interview about your solution.

Instructions

- Our stack is: Kotlin, Java, React, Spring Boot. Using languages / frameworks from our stack is a plus, but if you don't know these technologies, you are free to choose any programming language, library, tool or framework you prefer, as long as it's open source and freely available;
- Don't use libraries or any third-party code to solve the core of the exercise;
- It's ok to use libraries for UI, persistence, Web, JSON parsers, and anything not directly related to the given task;
- Every exercise must be done in a ready to execute format, so that is possible to build, run and see actual results;
- You can develop the code of the exercises in any form or platform (Desktop App, Web, Android, iOS, command line);
- You can merge multiple exercises in one single application/executable, as long as it's easy to identify the individual exercises;
- You **can write** your comments and documentation **in portuguese** (writing in english is a plus);
- Submit the results as a **Github repository**, created for this exam. If you prefer to keep the repository private, please share it with `kaffa-recruitment` Github user.
- Use the same repository for all the exercises;
- You can ask any question by email (in portuguese or english).

Deadline

One week (7 days).

The sooner, the better.

Evaluation criteria

- Quantity of completed exercises
- Complexity of the completed exercises
- Correctness of the solution
- Presentation of the content
- Code organization
- Techniques and practices
- Time to completion

Exercises

1) Validate CNPJ format and check digits

Given a string, check if it looks like a CNPJ, considering two formats:

- Formatted:

  ```
  "00.000.000/0000-00"
  ```

- Number only:

  ```
  "00000000000000"
  ```

Validate if it's a well-formed CNPJ, considering the "check digits", as defined by *Receita Federal*.

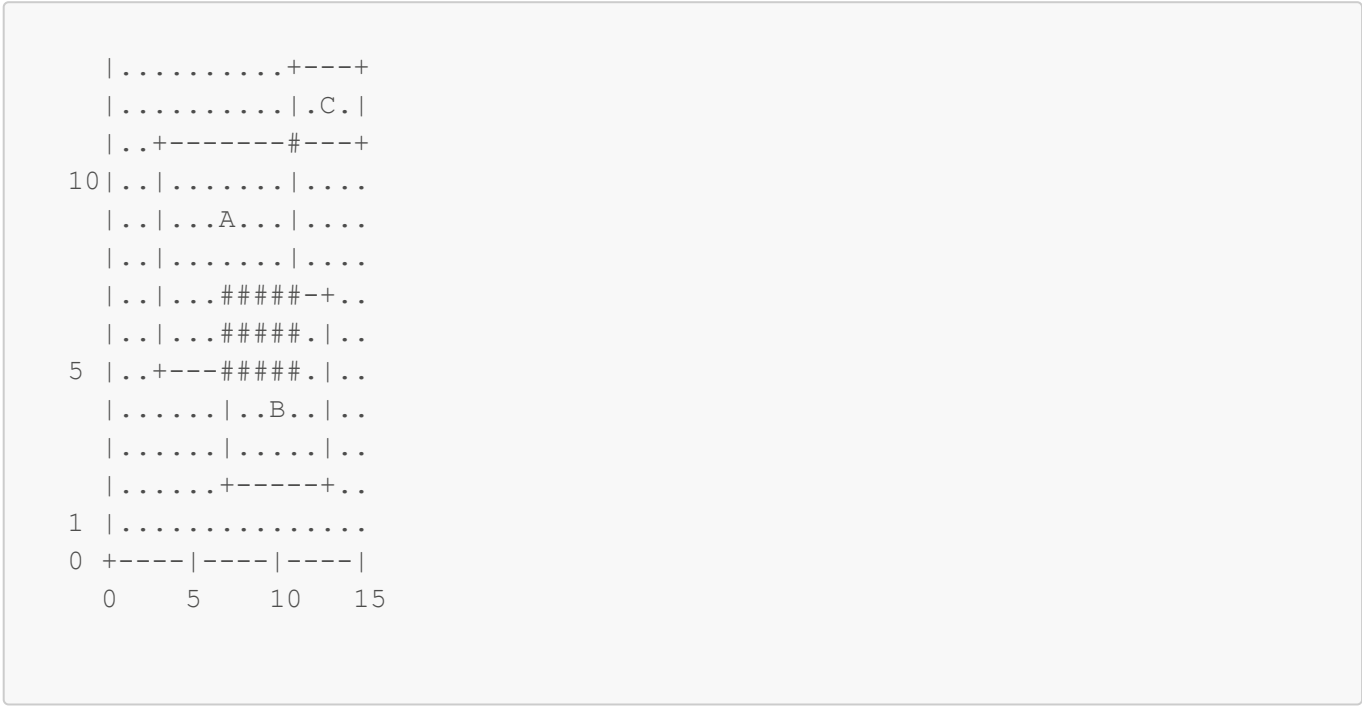
Important: *Don't use a library. You should write the validation code.*

2) Test if two rectangles intersect

Considering two rectangles in a discrete grid (like pixels in a display), each defined by two points, return true if they intersect, false otherwise.

Note: *the points are **included** in the rectangle and have a dimension of 1 unit; the rectangle (0, 0; 1, 1) have an area of 4 units.*

Example:




```
A = (3, 5; 11, 11)
B = (7, 2; 13, 7)
C = (11, 11; 15, 13)

intersects(A, B) => true
intersects(A, C) => true
intersects(B, C) => false
```

3) Compute the area of intersection between two rectangles

Considering two rectangles in a discrete grid (like pixels in a display), each defined by two points, compute the area of intersection between the two.

Note: *the points are **included** in the rectangle and have a dimension of 1 unit; the rectangle (0, 0; 1, 1) have an area of 4 units.*

Example:

```
    |..........+---+
    |..........|.C.|
    |..+-------#---+
  10|..|.......|....
    |..|...A...|....
    |..|.......|....
    |..|...#####-+..
    |..|...#####.|..
  5 |..+---#####.|..
    |......|..B..|..
    |......|.....|..
    |......+-----+..
  1 |...............
  0 +----|----|----|
    0    5    10   15
```

```
A = (3, 5; 11, 11)
B = (7, 2; 13, 7)
C = (11, 11; 15, 13)

areaOfIntersection(A, B) = 15
areaOfIntersection(A, C) = 1
```

4) Simple Todo List

Todo list application that permits the creation and deletion of tasks (texts).

- The application must persist the tasks between executions;
- Use any storage you want: database, files, PaaS backends (Firebase, etc.);

5) Rest Client - World Clock

Application that queries a server and displays the current date/time hour in local and UTC timezones.

Server URL: http://worldclockapi.com/api/json/utc/now

6) Rest Server - World Clock

REST server returning a JSON like:

```
{
    "currentDateTime":"2019-08-12T14:40Z"
}
```

7) Entity Relationship Diagram - Simple Order Manager

Design the model of a simple Order Manager System.

The system consists of:

- Clients
- Products
- Orders
- Any other tables you may need

You can draw, describe, or list the tables as SQL.

Extras:

- SQL: list ORDERS with number of items
- Which indexes should be created in this model?

Note: this exercise is documentation only - there's no executable to run in this case.

8) UX - Prototype (Optional - Designers / frontend developers only)

Suppose you are working on an app that uses maps for managing and designing new electric networks (poles, wire segments, transformers, substations...). The user of your app needs to create a project in order to design new elements in the electric network.

Design and prototype a screen for creating new projects. Each project needs to have the following informations, provided by the user:

- Project name (text)
- Project deadline (date)
- Project location. The user has to point a location on the map

Note: You can use any tools you want for drawing and prototyping, even pencil and paper. The result of this test **does not** need to be an interactive prototype. You can share just screenshots or pictures of your design.