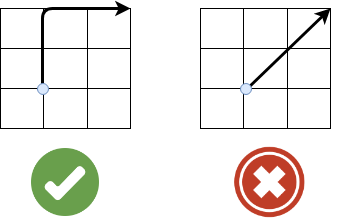
# **Part 1 - Coding exercise (2 problems: 45')**

## Problem 1

Duration: <5' (warm-up)

Create a function:

* Given 2 points in a bidimensional plane calculate the distance between them considering only ortogonal moves (no diagonal)



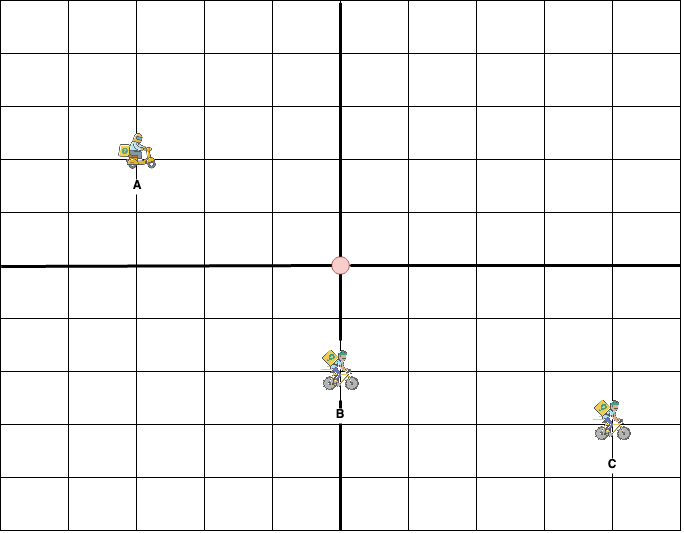
## Problem 2

Duration: 40'

Create a self-contained application, we recommend starting with the structure and signatures and progress with implementation bit by bit.

In Glovo we need an internal program to allocate orders to couriers and we want to prototype in the Test Town of 10 square metric units. Reusing the previously created functions and considering the following requirements, build the program to do so:

* An order may (or may not) be generated each unit of time
* Orders will have a pickup point and a delivery point
* Orders will only be assigned to idle couriers
* Couriers will only take one order at a time
* Couriers will move at the specified velocity when they have an assigned order (otherwise they will remain stopped)



### Test scenario:

* Given couriers:
  + A: position -3, 2; velocity: 3
  + B: position 0, -2, velocity: 2
  + C: position 4,-3, velocity: 2
* New orders at unit of time:
  + Time 1: pickup: 3,-2, delivery: 3,-4
  + Time 2: pickup: 3,-2, delivery: 2,-1
  + Time 3: pickup: 3,-2, delivery: 4,-3
  + Time 4:
  + Time 5: pickup: 1,3, delivery: 3,3

# **Part 2 - Data Processing (45')**

## Write a reusable data processing job to compute some KPIs based on the following input sources:

1. orders: JSON files in data/orders.json

2. geography: table public.geography in local PostgreSQL instance

The job should be written to an output table in the DB and should contain, for each date and country:

1. the number of orders

2. the total value of successful orders (in EUR)

3. the average order value (in EUR)

The job should be runnable both for a single day and for all the days available

Using the produced table, find the top 3 countries with the highest orders value on June 26, 2021.

## Initialize code:

import os

import requests

import psycopg2

from pyspark.sql import SparkSession

base\_remote\_path = "https://glovo-de-interview-0d7aa1a4.s3-eu-west-1.amazonaws.com/data\_processing"

manifest\_url = f"{base\_remote\_path}/manifest"

data\_urls = requests.get(manifest\_url, allow\_redirects=True).content.decode("utf-8")

base\_local\_path = "data"

for url in data\_urls.splitlines():

content = requests.get(url, allow\_redirects=True).content.decode("utf-8")

local\_path = url.replace(base\_remote\_path, base\_local\_path)

print(local\_path)

os.makedirs(os.path.dirname(local\_path), exist\_ok=True)

with open(local\_path, 'w+') as f:

f.write(content)

DB\_HOST = "postgres"

DB\_PORT = "5432"

DB\_USER = "postgres"

DB\_PASSWORD = "postgres"

DB\_SCHEMA = "public"

with psycopg2.connect(host=DB\_HOST, port=DB\_PORT, user=DB\_USER, password=DB\_PASSWORD, dbname=DB\_SCHEMA) as connection:

with open(f"{base\_local\_path}/init.sql") as f:

sql = f.read()

with connection.cursor() as cursor:

cursor.execute(sql)

db\_properties = {

'url': f'jdbc:postgresql://{DB\_HOST}:{DB\_PORT}/{DB\_SCHEMA}',

'user': DB\_USER,

'password': DB\_PASSWORD

}

spark = SparkSession.builder.appName("Test Pyspark").getOrCreate()

print(spark.read.format("jdbc").options(\*\*db\_properties, dbtable='geography').load().count())

print(spark.read.json("data/orders.json").count())

with psycopg2.connect(host=DB\_HOST, port=DB\_PORT, user=DB\_USER, password=DB\_PASSWORD, dbname=DB\_SCHEMA) as connection:

with connection.cursor() as cursor:

cursor.execute("SELECT count(\*) FROM geography;")

for r in cursor.fetchall():

print(r[0])