Code Test: Trip Sorter

Estimated resolution time: 3 hours

**Instructions**

First, to participate in this event, you must apply for this job position first. To apply you will have to fill in the form including your Escuela 42 identifier.

Carefully read the document. You will have to make your first decision. Do I want to be a Software Developer, QA, or do I not care?

If you choose to be a Software Developer, please read the Software Developer epigraph, for QA read Quality Assurance, and otherwise read both.

Software Developer (SD)

You have 3 hours to complete the test and you can use **any** programming language or framework you want.

No UI or database access code is needed. It is not mandatory to read input from file system. It is acceptable to read from stdin, forms or any other source.

This task is designed to give us an idea of how you think when faced with a very limited amount of time to solve a task of significant complexity.

We are interested in how you structure your code so that it's easily extendable, complies with best practices for the language used, and is easy to modify / understand by others.

We are also interested in seeing how efficient the algorithm you implement is. Feel free to include your complexity analysis in terms of time and/or space, use Big-O notation if possible.

Quality Assurance (QA)

You have 3 hours to complete the exercise

For a QA role one of the most valuable duties is that all the use cases are covered by one or more test cases (the more the better). For this task, you have to think those test cases, what they do and what are their inputs and output.

For example, for testing the divide operation in a calculator device you could write some tests like these:

• Test1 à Check specific result

• Input: dividend=25, divisor=5; Output:5

• Test2 à Check division by zero

• Input: dividend=25, divisor=0; Output: Error

Once you write all the test cases try to code some tests with your favorite programming language. We strongly recommend you code at least one or two test cases, but in case you have time feel free to write as many as possible.

Example:

• Python:

def divide\_operation(dividend, divisor):

try:

return dividend/divisor

except ZeroDivisonError:

return 0

• Test1

assert(divide\_operation(25, 5)==5)

• Test2

assert(divide\_operation(25, 0)==0)

This task is designed to give us a brief idea on how your thinking is facing a task of significant complexity with a considerable time restriction.

We are interested in how you structure your code so that it's easily extendable, complies with best practices for the language used, and is easy to modify / understand by others.

**Task**

You are given a stack of boarding cards for various transportations that will take you from a point A to point B via several stops on the way. All of the boarding cards are out of order and you don't know where your journey starts, nor where it ends. Each boarding card contains information about seat assignment, and means of transportation (such as flight number, bus number etc).

Provide an API that let's you sort this kind of list and present back a description of how to complete your journey. For instance the API should be able to take an unordered set of boarding cards, provided in a format defined by you, and produce this list:

• Take train 78A from Madrid to Barcelona. Sit in seat 45B.

• Take the airport bus from Barcelona to Gerona Airport. No seat assignment.

• From Gerona Airport, take flight SK455 to Stockholm. Gate 45B, seat 3A. Baggage drop at ticket counter 344.

• From Stockholm, take flight SK22 to New York JFK. Gate 22, seat 7B. Baggage will we automatically transferred from your last leg.

• You have arrived at your final destination.

The list should be defined in a format that's compatible with the input format.

**Requirements**

• Use object-oriented language if it is possible

• Do not use any 3rd party framework. Start all code from scratch.

• The structure of the code should be extendable to make building in support for any means of transportation / extra information required about a specific type of transportation easy.

• The implementation of your sorting algorithm should work with any set of boarding passes, as long as there is always an unbroken chain between all the legs of the trip. Ie. it's one continuous trip with no interruptions.

• The algorithm doesn't need to consider that departure / arrival are in the correct order. In fact, there is no information about any such times on the boarding passes. It is just assumed that your next connection is waiting for you when you arrive at a destination.

• The algorithm should have the lowest possible order of complexity (Big O notation) you could think of.

**Deliver**

Deliver your solution along with any notes, comments, and assumptions you have made while working on the solution via e-mail to the recruiter who sent you this test.

Usually, mail clients block executable files or with code, so, for sharing the code with the team, a cloud tool should be used. We need a **PUBLIC** link (reviewers will download it) with the zip file. If you do not have services like google drive, we recommend https://wetransfer.com/, with a couple of clicks you can get a public download URL with your ZIP file.

Do not publish the problem description or the solution you implement.

**Example E-mail for Software Developer**

**To**:

**Subject**:[MVP42][SD] Prueba YYYYYY

**Body:**

Hi!

My name is XXXXXX and my Escuela 42 identifier is YYYYYY.

Here is the link for the code that I have developed https://ZZZZZZ

Regards!

**Example E-mail for QA**

**To**:

**Subject**: [MVP42][QA] Prueba YYYYYY

**Body:**

Hi!

My name is XXXXXX and my Escuela 42 identifier is YYYYYY.

Here is the link for the code that I have developed https://ZZZZZZ

Regards!

**Example E-mail for both**

**To**:

**Subject**: [MVP42][ALL] Prueba YYYYYY

**Body:**

Hi!

My name is XXXXXX and my Escuela 42 identifier is YYYYYY.

Here is the link for the code that I have developed https://ZZZZZZ

Regards!

Please replace the fields XXXXXX, YYYYYY and ZZZZZZ with your name and surname, your Escuela 42 identifier, and the link to your code test zip file.