Coding Test

Introduction

The goal of this test is to be able to check the coding skills in an environment more open than a coding game or a technical interview. As such it'll require you to create a full project to solve a problem.

Python being our main language, the test is written for Python, however if you're not confortable with Python contact us and we can agree to use another language.

Evaluation Criteria

As the test focuses on creating a full project rather than writting a short function the evaluation criteria will be broader than checking is the optimal answer is provided.

In random order those criteria will include:

- Easiness of build/install/run
- Code or packaging quality
- Correctness of the answer (good answer even if not optimal)
- Speed and memory usage
- Creativity
- Optimal answer

Test: the Spaceship Rental Problem

Description

You're going to optimize the profitability of a small rental company with a single spaceship to rent. You'll receive a list a contracts, each order will include:

- a name (String of 64 chars or less)
- a start hour (int)
- a duration (int)
- a price (int)

Start hour, duration and price are all positive integers that can quite large.

The goal is to produce a list of contract names maximizing the profit. As there's only a single ship to rent there should not be any overlap between the accepted contracts and of course not all proposed contracts can be picked.

For instance we can have a 4 contract list:

- Contract1: start hour 0, duration 5, price 10
- Contract2: start hour 3, duration 7, price 14
- Contract3: start hour 5, duration 9, price 8
- Contract4: start hour 5, duration 9, price 7

Optimal solution would be *Contract1* and *Contract3* with a total income of 18. Another solution would be *Contract1* and *Contract4* which will also lead to a continuous rental from 0 to 14 but would only give a 17 income.

What To Do?

The goal is to create a small project that'll include an application to solve the *spaceship rental problem*.

Application

A small webserver is the prefered way to interact.

When started, your application should create a simple web server listening on port `8080` on all interfaces. The only end point required is `/spaceship/optimize` accepting `POST` requests with a JSON payload containing the list of contracts (see below for the format). The server should compute a solution and return a JSON object with the total income and the list of contract names (in order) to achieve this income.

Payloads

Input

The input payload will be like:

```
[
    {"name": "Contract1", "start": 0, "duration": 5, "price": 10},
    {"name": "Contract2", "start": 3, "duration": 7, "price": 14},
    {"name": "Contract3", "start": 5, "duration": 9, "price": 8},
    {"name": "Contract4", "start": 5, "duration": 9, "price": 7}
]
```

Output

The server should return a payload similar to the following:

```
{
    "income": 18,
    "path": ["Contact1", "Contract3"]
}
```

Requirements

1. You **should** send your source code in a zip archive (tar.gz accepted).
2. You **should** include a small documentation detailling how to package, install and execute your application.
3. The target language is Python 3.10 and everything (build, packaging, execution) **should** be doable on a clean Python 3.10 environment.
4. The application **should** work on both Windows and Linux

5. The webserver is the prefered implementation but falling back on a simple CLI script is OK provided that:
    1. The script accepts/returns the payload described above
    2. The script usage is documented
6. No pre-compiled binary will be accepted