Minnovi Backend Take-Home Assignment

For this assignment you will be asked to write a simple API that handles in a, we hope, fun way some not so complex data.

Your API will receive a JSON payload containing two entities: 'myself' and 'enemy' and will return the result of a 'duel' between them. For simplicity's sake a 'duel' is the damage dealt by the entity called 'myself' hitting with its weapon the entity called 'enemy'.

The input

First, the would-be frontend of this application makes the player fight against an enemy. This situation is sent to your API with a JSON payload like this example:
We will analyze the following JSON, don't worry.


```
{
    "data": {
        "duel": {
            "enemy": {
                "name": "The Mute Musician",
                "headArmour": {
                    "name": "Obsidian Jaws of Infinite Ancestors",
                    "defence": {
                        "physical": 10,
                        "lightning": 10,
                        "fire": 10
                    }
                },
                "chestArmour": {
                    "name": "Ghastly Ebon Chestplate",
                    "defence": {
                        "physical": 20,
                        "lightning": 20,
                        "fire": 20
                    }
                },
                "weapon": {
                    "name": "Atuned Trinket",
                    "attack": {
                        "physical": 100,
                        "lightning": 50,
                        "fire": 10
                    }
                },
                "talents": [
                    "Massive",
                    "God",
                    "Fire affinity"
```

```
            ]
        },
        "myself": {
            "name": "Hall",
            "headArmour": {
                "name": "Plate Casque of Unholy Lands",
                "defence": {
                    "physical": 10,
                    "lightning": 10,
                    "fire": 10
                }
            },
            "chestArmour": {
                "name": "Wind-Forged Chestpiece of Stealth",
                "defence": {
                    "physical": 20,
                    "lightning": 20,
                    "fire": 20
                }
            },
            "weapon": {
                "name": "Earthfire",
                "attack": {
                    "physical": 100,
                    "lightning": 50,
                    "fire": 10
                }
            },
            "talents": [
                "Fire affinity",
                "Safety first",
                "God"
            ]
        }
    }
}
```

Entity

There will be two entities: myself and the enemy. Each entity will have the following structure:

- Name: a string containing the name of the entity
- Head armour: an object containing the name and the defence stats
- Chest armour: an object containing the name and the defence stats
- Weapon: an object containing the name and the attack stats
- Talents: an array of talents (there will always be at least 1 talent and a max of 3 talents)

Defence stats

Each piece of armour has 3 types of defence: physical, lightning and fire each represented with an integer greater than 0. These stats will be used to compute the result of the duel.

Attack stats

Every weapon has 3 types of attack: physical, lightning and fire, each represented with an integer greater than 0. These stats will be used to compute the result of the duel.

Talents

Talents are special attributes that modify attack and defence stats.
Here are all the talents:

- Fire Affinity: +5 damage done by fire
- Lightning Affinity: +5 damage done by lightning
- God: +10 def on all armour pieces on elemental stats
- Massive: +5 physical defence on chest armour
- Light Weight: -5 physical defence on all pieces of armour but +10 fire defence on them
- Blacksmith: +10 defence on every defence stat on chest armour
- Safety First: +7 defence on everything on head armour
- Almighty: +50% of damage on every attack stat

Calculating the damage

First of all, calculate the effects of the talents (if any).
Then calculate the 'raw damage'. This consist of the sum of all damage type of your weapon. Next calculate the 'effective damage'. The defence stats are actually damage mitigation in percentage. You will have to return a JSON with the effective damage of each attack/defence stats following the formula
`(damage - (chest defence)%) - (head defence)%`

Example

Let's make an example so everything is clear, given the input seen above:

1. Calculate the 'raw damage' of your weapon: 100 + 50 + 10 = 160
2. Calculate the effect of the talents on 'myself' and 'enemy'. Talents must be calculated in the given order of the array
3. Calculate the 'effective damage' of your weapon.
   In our example, for physical 'effective damage': (100 - 25%) - 10% = 67.5 = 68
   The same thing has to be done for fire damage and lighting damage.
4. Return the values in a JSON formatted exactly like this:

```
{
  "enemy": {...},
  "myself": {...},
  "rawDamage": 160,
  "effectiveDamage": {
    "physical": 68,
    "fire": x,
    "lightning": y,
  }
}
```

Endpoints

In order to get the data just send a GET request to https://hiring-test-dxxsnwdabq-oa.a.run.app/duel
Send the processed data with a POST to https://hiring-test-dxxsnwdabq-oa.a.run.app/processDuel
The body of the POST request must be

```
{
  "data": {
  "enemy": {...},
  "myself": {...},
  "rawDamage": 160,
  "effectiveDamage": {
    "physical": 68,
    "fire": x,
    "lightning": y
    }
  }
}
```

What's the point?

The purpose of this test is to assess your approach and thought process in completing it. While the test itself is not inherently difficult, it provides an opportunity to showcase your chosen strategies. We will evaluate your project based on the following criteria:

1. Readability: We will assess how easily understandable your code is, including the presence of comments.
2. Modularity: We are interested in whether your code is structured as a single, large function or if you have divided it into smaller, more manageable functions.
3. Future scalability: Consider the scenario where the Project Manager wants to introduce additional elements, such as another piece of armor. We will evaluate how efficiently this modification can be made, either by you or another developer.
4. Optimization: While not a primary focus, we encourage you to avoid exponential complexity in your code. However, we will not solely judge you based on code optimization or the length of your functions. By considering these aspects, we aim to gain insights into your problem-solving skills and coding practices.

Should you use a specific framework?

No. Use what you feel most comfortable with.

Questions?

For any question, even the ones you consider *silly* , do not hesitate to contact me at andrei.gorgan@minnovi.it