ID1019: Advent of code 2022 day 1

Alexander Lundqvist

Spring Term 2023

Introduction

In this assignment we are trying to solve Advent of Code 2022 day 1. In the assignment we have a number of elves who carry different amounts of food items in their backpacks and are represented by their calories. We assume that the elves and their inventory is represented as a list in a separate text file where each elf separates their own inventory from the previous elf's inventory by a blank line. The task is to find the elf that carries the most calories.

Method

We start by creating inventory as `data.txt` and insert the example list described the assignment instructions. The actual solution is defined with a function `solve/1` which takes a path to the inventory file as its argument. To handle the input from the text file we also define a function `parse/1` which will process the file and produce a `Map` where the keys will be sequential and the values are arrays with integers. Lastly, to facilitate easier testing of the program, we define a `run/0` function that handles the whole execution including specifying the file path. Technically we could also hard code a static inventory to make the module self contained.

Result

The solution can in theory be performed with only one function, but as this generally is bad coding practice, the code has been separated into multiple single purpose functions (I assume that functional programming languages as Elixir adheres to the common programming concepts cohesion, coupling and encapsulation). With that said, I think it is only necessary to show one of the functions as they do look quite similar. Below is the parsing function that reads from a specified path, then pipes the return value to the `String.split/2` to separate the individual elves and then pipes the result

to the `Enum.map/1` to assign each elf an index or ID which is connected to their respective "backpack".

```
def parse_inventory(file_path) do
    File.read!(file_path)
    |> String.trim()
    |> String.split("\n\n")
    |> Enum.map(&String.split(&1, "\n"))
end
```

Executing the code gives us this output which we can verify by the intended outcome described in the assignment.

```
iex(2)> Day1.run()
Welcome to day 1 of Advent of Code 2022!
The elves default inventory is:
1000
2000
3000

4000

5000
6000

7000
8000
9000

10000

The elf that carries the most calories is:
{24000, 4}
```

The entire code is available at this github repository.

Discussion

I don't know what there is to discuss about the assignment in itself and I must admit that this wasn't my most ambitious work so far since these last two weeks have been hectic with the preliminary work for my bachelor's thesis.

My solution is inspired by a friends who based theirs on piping and since I've only really worked with a recursion based approach so far I felt like I

should give it a go. I have had previous experience with the concept from using bash as well as from the course EN2720 Ethical Hacking.

The current implementation is quite simple as it has no error handling, extensive testing for reliability nor checks for any malformed input. Since this was more or less an assignment that we ourselves could define the scope of, I had a hard time justifying spending more time on it. I think I actually prefer when we test, compare and run benchmarks.