Task 5 Advent of Code Day 1

Adrian Jonsson Sjödin

Spring Term 2023

Introduction

In this report we solve the day one puzzle of Advent of Code 2022. The task was to find the elf carrying the most calories among a group of elves and return which elf it was and how many calories they where carrying in total. As input we have the elves inventory where every elf's item's calorie count is listed on a separate line, and each el's inventory is separated by a blank line.

Method

The solution for the code consists of two functions. The first function, `parse/1`, takes the input read from the file as a string and trims it, splits by the blank line to separate by elf and then converts each elf's inventory from a list of strings to a list of integers.

The second function, `solve/1`, is the one that will be called from `iex` to solve the problem. It takes the input file path as an argument, reads the content and passes it to the `parse/1` function. It then maps each elf's inventory that `parse/1` produce to a list containing the total calorie count. It also uses `Enum.with_index(1)` to add the elf's index to the list. The 1 in the argument is because we want to start counting from one and not zero. Finally the last step is to return the list with the highest total calories count and the index for the elf who had it.

Result

The code is tested with the text file seen in code overview 1 and produces the output seen in code overview 2. It means that the elf with the highest calorie count is elf number 1 with a calorie count if 26,000 calories in its inventory. Doing a manual check against the file we see that this is indeed correct.

Code Overview 1: inventory.txt

```
1000
2000
3000
20000

4000

5000
6000

7000
8000
9000

10000
```

Code Overview 2: Result of running AOC.solve("inventory.txt")

```
{[26000], 1}
```

Discussion

The challenge with this task was to understand how to correctly use the pipe operator `|>` and the `&` symbol which is used to create an anonymous function capture. This was needed to be able to pass the result from one operation to the next. It did however help by thinking of them as similar to how you can chain functions calls together in JavaScript, and how you use the pipe operator in bash.

The solution itself is quite straightforward and unlike previous task did not require any recursion.

One drawback of this solution is that it assumes that the file we read from have all data in the correct format. We do not check for any potential errors or do any form of error handling. However this was not part of the task so I think this solution is justified.

Another thing that could be improved is the reporting of more relevant information, like the total number of elves or the size of each elf's inventory.

The code in its entirety can be found here: GitHub.