Day 1 Advent of Code 2022

Axel Månson Lokrantz

Spring Term 2023

Introduction

The task for this assignment was to tackle the day one problem of advent of code 2022 through functional programming in Elixir. The program was written together with Daniel Dahlberg at KTH.

Problem 1

A number of elves are on an expedition, each elf has a given set of calories on them. In a text file the elves rations are written down. One item per line. Each elf separates their own inventory from the previous elf's inventory (if any) by a blank line.

```
500
100

200
100
```

In the example above, the first elf has an inventory of $500 + 100$ calories, and the second elf an inventory of $200 + 100$. The first task is to find the elf carrying the most calories and how many total calories that elf is carrying. To be able to solve the problem we first had to import the raw text file.

```
rawFile = File.read!("inputs/d01")
processedFile = String.split(rawFile,"\n")
list = create_list(processedFile, 0, [])
```

The arguments our function takes is the raw text file split into segments, an accumulator to be able to sum each elf's total calories and a list where we store the new values. In all cases except for when our head is equal to a new line (marked as an empty string `""`) the program translate the the input to an integer and adds it to the accumulator. If the head equals to an

empty string, the value of our head is set to the value of our accumulator and the accumulators value is set to 0. The entries in the list now represent the total calories each elf is carrying. To find the elf with the most calories we used the built in `Enum.max` function which return the largest integer of a list.

```
def create_list([], acc, lst) do [acc | lst] end
def create_list([h | t], acc, lst) do
    case h do
      "" -> [acc | create_list(t, 0, lst)]
      _  ->
      {h, _} = Integer.parse(h)
      create_list(t, acc + h, lst)
    end
  end
```

Problem 2

Problem 2 was then to find the top three elves carrying the most calories and how many calories those top three elves carried in total.

```
def top_three([], f, s, t) do [f, s, t] end
  def top_three([h|tail], f, s, t) do
    cond do
      h > f ->
        top_three(tail, h, f, s)
      h > s ->
        top_three(tail, f, h, s)
      h > t ->
        top_three(tail, f, s, h)
      true ->
        top_three(tail, f, s, t)
    end
  end
```

This was solved through a simple function which check for certain conditions. The function takes a list, and three variables `f` (first), `s` (second), and `t` (third) which represent the three largest integers in the list. If the head is larger than `f` we rearrange the values, `f` takes the value of the head, `s` takes the value of `f` and `t` the value of `s`. Similarly if h is smaller than `f` but larger than `s`, the value of `f` remains the same and `s` and `t` are updated and the function is called recursively. To sum the list of the total calories carried by the top three elves we used `Enum.reduce`.

```
Enum.reduce(top_three(list, 0, 0, 0), fn(x, acc) -> acc + x end)
```