Advent of Code

Docker image runner

December 1, 2022

Contents

1 Day 1

1.1 Emacs Lisp

```
(defun sorted-calories (input-file)
  "Read the file and return a sorted list of elf item calories."
  (with-temp-buffer
    (insert-file-contents input-file)
    (let* ((file-content (buffer-string))
   (list (split-string file-content "\n\n"))
   (list-of-lists (mapcar
   (lambda (list) (split-string list "\n"))
   list))
   (parsed (mapcar (apply-partially #'mapcar #'string-to-number) list-of-lists))
   (summed-up (mapcar (apply-partially #'cl-reduce #'+) parsed)))
      (sort summed-up #'>))))

(let* ((sorted (sorted-calories "/tmp/input.txt"))
       (part-1-solution (car sorted))
       (part-2-solution
(apply #'+ (seq-take sorted 3))))
  (print (list part-1-solution part-2-solution)))
```

1.2 Haskell

```
import Data.List (sort)
import Data.List.Split (splitOn)

sortedCalories :: String -> [Int]
sortedCalories =
    (reverse . sort . (map sum) . (map (map read)) . (splitOn [""]) . lines)

highest = head . sortedCalories

top3highest = sum . (take 3) . sortedCalories

main = do
  file <- readFile "/tmp/input.txt"
  -- Part 1 solution
  print . highest $ file
  -- Part 2 solution
  print . top3highest $ file
```