TPFI 2022/23
Hw 4: Effectful Haskell

assegnato: 10 maggio 2023, consegna 22 maggio 2023

Esercizio 1 (Input/Output) Definite un'azione `charCount :: IO ()` che legge un numero n da tastiera, poi n stringhe e alla fine stampa il numero di stringhe in cui appare ciascuna lettera.

Esercizio 2 (Applicativi) Un numero n è *perfetto* se è uguale alla somma dei suoi divisori propri (compreso l'1 ed escluso n). Ad esempio, il $28 = 1 + 2 + 4 + 7 + 14$. Tutti gli altri numeri sono classificabili come *difettivi* (se la somma dei divisori è minore del numero, ad esempio tutti i primi sono difettivi) oppure *abbondanti* (se la somma dei divisori è maggiore del numero, ad esempio il $24 < 1 + 2 + 3 + 4 + 6 + 8 + 12 = 36$).

Tra i numeri abbondanti alcuni sono detti *semi-perfetti* perché sono uguali alla somma di un sottoinsieme dei loro divisori. Ad esempio il $24 = 1 + 2 + 3 + 4 + 6 + 8$ oppure $24 = 1 + 2 + 3 + 6 + 12$ o anche $24 = 4 + 8 + 12$.

Provare a verificare che un numero è semiperfetto generando tutti i sottoinsiemi dei divisori usando `<*>` sulle liste (che "moltiplica" le computazioni) e poi filtrando solo i sottoinsiemi con la somma uguale al numero (magari usando il tipo `Maybe`).

Esercizio 3 (Monadi I) Definire il tipo `Either` come istanza di `Monad` (osservare che `Maybe` è un caso particolare di `Either`).

Esercizio 3 (Monadi II) Usare la monade `Either` del tipo precedente per estendere il valutatore con eccezioni di espressioni aritmetiche visto a lezione. Considerare tutte e 4 le operazioni aritmetiche (+, ×, :, e -) e pensare le espressioni sul tipo `Nat` (cioè i naturali costruiti con 0 e `succ`). Dovete usare il tipo `Either` come segue: i valori sinistri sono valori corretti, mentre quelli a destra sono eccezioni, che possono essere di due tipi, generati dalla divisione (per zero) oppure dalla sottrazione (quando il secondo argomento è maggiore del primo).