# TPFI 2020/21
# Hw 4: Effectful Haskell

assegnato: 12 maggio 2021, consegna 24 maggio 2021

**Esercizio 1** (Input/Output) Definite un'azione `adder :: IO ()` che legge un numero $n$ da tastiera, poi $n$ numeri interi e alla fine stampa la loro somma.

**Esercizio 2** (Foldable I) Far vedere che il tipo `Maybe` può essere reso un'istanza della classe `Foldable` dando le definizioni esplicite delle funzioni `fold`, `foldMap`, `foldr`, `foldl` e `traverse`.

**Esercizio 3** (Foldable II) Come nell'esercizio precedente, ma per il seguente tipo di albero binario:

```
data BinTree a = EBT | Fork (BinTree a) a (BinTree a)
   deriving Show
```

**Esercizio 4** (Monadi I) Usualmente, la classe `Monad` viene definita come sottoclasse di `Applicative` che a sua volta è sottoclasse di `Functor`. Seguendo quest'ordine, quando vi ho definito il tipo state transformer `ST`, l'ho definito prima come istanza di `Functor`, poi di `Applicative` e infine di `Monad` (vedi slides Lezione **14**).

Supponendo però di aver definito prima `>>=`, dare delle definizioni che fanno uso di `do`-notation di `<*>` e `fmap` per il tipo `ST`. Tra l'altro, in Haskell, non è (in questo caso) rilevante l'ordine delle definizioni.

**Esercizio 5$^\star$** (Monadi II) Provate a parametrizzare il tipo `ST` su un generico tipo (non necessariamente `Int` come negli esempi visti a lezione). In caso di successo, fornire alcuni esempi di utilizzo.