Esame di Linguaggi e Paradigmi di Programmazione (6 CFU)

Regolamento

- Il tempo a disposizione per la risoluzione degli esercizi di laboratorio è di **20 minuti**.
- Non è consentita la consultazione di appunti, dispense, libri o l'uso di laptop, tablet, ecc.
- Per la risoluzione degli esercizi 1 e 2 è possibile usare un editor a scelta, l'ambiente interattivo `GHCi` e Hoogle sulla postazione di laboratorio occupata.
- Il tempo a disposizione per la risoluzione degli esercizi 3 e 4 è di **40 minuti**.

1 Laboratorio

Esercizio 1 (7 punti)**.** *Dato il tipo algebrico*

```
data Tree a = Empty | Node a [Tree a]
```

per rappresentare alberi n-ari, definire una funzione `elements :: Tree a → [a]` *che calcola la lista di tutti gli elementi contenuti nell'albero in un ordine a scelta. Usare la ricorsione solo laddove necessario, sfruttando il più possibile le funzioni del modulo* `Prelude`*. Se opportuno, è ammessa la definizione di funzioni ausiliarie.*

Esercizio 2 (7 punti)**.** *In riferimento al tipo algebrico* `Tree` *dell'esercizio precedente, diciamo che un albero è in forma normale se è* `Empty` *oppure se è costruito senza usare* `Empty`*. Definire una funzione* `normalize :: Tree a → Tree a` *che trasforma un albero in forma normale, usando la ricorsione solo laddove è necessario e sfruttando il più possibile le funzioni del modulo* `Prelude`*. Se opportuno, è ammessa la definizione di funzioni ausiliarie.*

2 Teoria

Esercizio 3 (8 punti)**.** *Applicare l'algoritmo di inferenza all'espressione*

```
foldr (.) (\x → x)
```

per determinarne il tipo più generale, dove

```
foldr :: (a → b → b) → b → [a] → b
(.)   :: (b → c) → (a → b) → a → c
```

Esercizio 4 (8 punti)**.** *Dimostrare la proprietà*

```
sorted xs == all (uncurry (<=)) (zip xs (tail xs))
```

dove

```
sorted :: Ord a ⇒ [a] → Bool
sorted [] = True
sorted [_] = True
sorted (x : y : xs) = x <= y && sorted (y : xs)
```

Indicare i principi di dimostrazione applicati e giustificare ogni passaggio della dimostrazione:

- *proprietà note delle operazioni aritmetiche (es. commutatività e associatività di + e *) possono essere assunte ma vanno comunque* ***menzionate****;*
- *eventuali riferimenti a funzioni di libreria (es. con* `foldr.1`*) vanno accompagnati dalla definizione* ***completa*** *della funzione (es. di* `foldr`*);*
- *eventuali altre proprietà utilizzate vanno* ***dimostrate*** *esplicitamente.*