Programmazione Funzionale – Febbraio 2021

Nota: è indispensabile specificare il tipo e dare una descrizione dichiarativa di ogni funzione ausiliaria utilizzata (anche locale), altrimenti non verrà presa in considerazione (ad eccezione delle funzioni il cui tipo e specifica sono già dati nel testo).

1. Scrivere un programma con una funzione `filquad: int list → int list → int list`, tale che `filquad ints quads` restituisca una lista contenente, in qualsiasi ordine, tutti e solo gli elementi della lista `ints` il cui quadrato è presente in `quads`. Ad esempio, la funzione, applicata alla lista `[1;2;3;4;5]` e `[9;25;10;4]` riporterà una lista contenente gli elementi 2, 3 e 5.

   Scrivere diverse versioni della funzione:

   (a) una che non utilizzi la funzione `List.filter` e implementi un algoritmo ricorsivo;

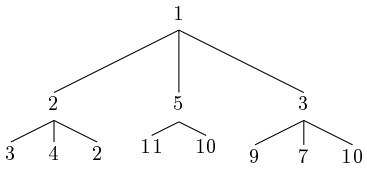
   (b) una che non utilizzi la funzione `List.filter` e implementi un algoritmo iterativo;

   (c) una che utilizzi la funzione `List.filter`.

2. In un albero etichettato da interi, il peso di un ramo è la somma delle etichette dei nodi che compaiono su quel ramo.

   Definire un tipo `'a ntree` per la rappresentazione di alberi n-ari e una funzione `pesi: 'a ntree → 'a list` che, applicata a un albero `t`, riporti una lista contentente tutti i pesi di tutti i rami che vanno dalla radice a una foglia in `t`.

   Ad esempio, se `t` rappresenta l'albero illustrato in figura, `pesi t` riporterà una lista contenente (in qualsiasi ordine) gli interi 6, 7, 5, 17, 16, 13, 11 e 14.



3. Descrivere che cosa si intende per forma currificata di una funzione e quali vantaggi si possono avere nel definire funzioni in forma currificata. Per illustrare quanto si afferma, introdurre e discutere un esempio di propria scelta.