Programmazione funzionale (OCaml)

https://taroccoesbrocco.github.io/progfunz.html

Esame — 8 luglio 2025

Esercizio 1 (15 punti)

Si consideri la funzione `List.exists : ('a -> bool) -> 'a list -> bool` tale che `List.exists p lst` restituisce `true` se e solo se almeno un elemento della lista `lst` soddisfa il predicato `p` (in particolare, restituisce `false` se `lst` è vuota). Si definiscano le funzioni seguenti aventi lo stesso tipo e comportamento di `List.exists`:

1. (5 punti) `exists_rec`, che è ricorsiva (nel senso che utilizza la parola chiave `rec` nella sua definizione) e *non* usa alcuna funzione del modulo `List`;
2. (5 punti) `exists_fold`, che *non* è ricorsiva (nel senso che non usa la parola chiave `rec` nella sua definizione) e utilizza `List.fold_left` o `List.fold_right`, ma nessun'altra funzione del modulo `List`;
3. (5 punti) `exists_lib`, che *non* è ricorsiva (nel senso che non usa la parola chiave `rec` nella sua definizione) e usa una qualsiasi combinazione delle funzioni del modulo `List` diverse da `List.fold_left` e `List.fold_right`.

Esercizio 2 (20 punti)

Si consideri la struttura dati per gli alberi binari con nodi di tipo parametrico `'a`, definita a lezione come

```
type 'a tree = Empty | Tr of 'a * 'a tree * 'a tree
```

Un *albero binario di ricerca* è un albero binario `t` di tipo `'a tree` tale che nel tipo `'a` è definito un ordine e, per *ogni* nodo `n` di `t`, i nodi nel sottoalbero alla sinistra di `n` sono tutti strettamente minori di `n` ed i nodi nel sottoalbero alla destra di `n` sono tutti strettamente maggiori di `n` (in particolare, l'albero vuoto è un albero binario di ricerca).

1. (5 punti) Si definiscano delle funzioni `leftmost : 'a tree -> 'a` e `rightmost : 'a tree -> 'a` tali che `leftmost t` e `rightmost t` restituiscano, rispettivamente, il nodo più a sinistra e il nodo più a destra dell'albero binario (non necessariamente di ricerca) `t`. Se `t` è vuoto, le funzioni sollevano l'eccezione `Not_found`.
2. (8 punti) Si definisca una funzione `is_bst_with_min_max : 'a -> 'a -> 'a tree -> bool` tale che `is_bst_with_min_max minval maxval t` restituisca `true` se e solo se `t` è un albero binario di ricerca i cui nodi sono tutti strettamente maggiori di `minval` e strettamente minori di `maxval`.
3. (7 punti) Senza usare la ricorsione e utilizzando le funzioni `leftmost`, `rightmost` e `is_bst_with_min_max` dei punti precedenti, anche se non sono state definite, si definisca una funzione `is_bst_int : int tree -> bool` tale che `is_bst_int t` restituisca `true` se `t` è un albero binario di ricerca, `false` altrimenti.

   Suggerimento: Si faccia attenzione al valore massimo e al valore minimo dei nodi di `t` (che sono di tipo `int`).