Programmazione funzionale (OCaml)

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

Esame — 17 giugno 2025

Esercizio 1 (10 punti)

Si considerino le definizioni sequenti e si risponda alle domande qui sotto, giustificando ogni risposta.

```
let square x = x*x ;;
let twice f x = f (f x) ;;
let fourth = twice square ;;
let what = fourth 3 ;;
```

1. (5 punti) Qual è il tipo di `twice`? Qual è il tipo di `fourth`?
2. (5 punti) Quali sono il tipo e il valore di `what`?

Esercizio 2 (10 punti)

Si consideri il tipo polimorfo `'a option` di OCaml, visto a lezione, definito come

```
type 'a option = None | Some of 'a
```

Si definiscano le funzioni seguenti:

1. (2 punti) `succ_opt : int option -> int option` tale che `succ_opt n` restituisca:
   - `Some k` se `k` è il successore dell'intero `n`, cioè se `k = n+1`,
   - `None` se `n = None`.
2. (3 punti) `map_opt : ('a -> 'b) -> 'a option -> 'b option` tale che `map_opt f x` restituisca:
   - `Some y` se `y` è il valore restituito dalla funzione `f` quando è applicata a `x`,
   - `None` se `x = None`.
3. (5 punti) `max_opt : 'a list -> 'a option` tale che `max_opt lst` restituisca:
   - `Some n` se `n` è l'elemento massimale della lista `lst`,
   - `None` se invece `lst` è vuota.

Esercizio 3 (15 punti)

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

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

1. (10 punti) Si definisca una funzione `search_path : ('a -> bool) -> 'a tree -> 'a list` tale che `search_path p t` restituisca un cammino (se esiste, altrimenti solleva un'eccezione `NotFound`) dalla radice dell'albero binario `t` a un nodo di `t` che soddisfi la proprietà `p`. Il cammino è la lista dei nodi che portano dalla radice di `t` (che è la testa della lista) a un nodo che soddisfi la proprietà `p`.

   Suggerimento: Utilizzare la tecnica di *backtracking* vista a lezione e una funzione ausiliaria locale con un accumulatore per memorizzare il cammino.
2. (5 punti) Utilizzando la funzione `search_path` del punto precedente, anche se non è stata definita, si definisca una funzione `path_to : 'a -> int -> 'a list` tale che `path_to n k t` restituisca un cammino (se esiste, altrimenti solleva un'eccezione `NotFound`) dalla radice dell'albero binario `t` a un nodo `n` di lunghezza esattamente `k`. La lunghezza di un cammino è la lunghezza della lista dei suoi nodi. Si assume che l'albero binario `t` non abbia più nodi con la stessa etichetta.