Programmazione funzionale (OCaml)

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

Esame — 9 settembre 2025

Esercizio 1 (15 punti)

1. (5 punti) Si definisca una funzione `(@@) : ('b -> 'c) -> ('a -> 'b) -> 'a -> 'c` tale che `(@@) g f x` restituisca il valore della funzione `g` applicata alla funzione `f` applicata al valore `x`.

2. (5 punti) Date delle funzioni `g : 'b -> 'c` e `f : 'a -> 'b`, qual è il tipo dell'espressione `g @@ f`? Che valore viene restituito dall'espressione `(String.length @@ string_of_int) 100`? Si giustifichino brevemente le risposte.

3. (2 punti) Date delle funzioni `g : 'b -> 'c` e `f : 'a -> 'b` e un valore di tipo `lst : 'a list`, qual è il tipo dell'espressione `List.map g (List.map f lst)`? Si giustifichi brevemente la risposta.

4. (3 punti) Nelle stesse ipotesi del Punto 3 riguardo a `f`, `g` e `lst`, si scriva una espressione equivalente a `List.map g (List.map f lst)` che usi `List.map` una sola volta, utilizzando la funzione `(@@)` del Punto 1.

Esercizio 2 (20 punti)

Si consideri il tipo `date` definito come

```
type date = int* int * int
```

per rappresentare le date nel formato (anno, mese giorno).

1. (5 punti) Si definisca una funzione `is_date : date -> bool` tale che `is_date dt` restituisca `true` se `dt` è una data valida, `false` altrimenti. Una data `(d,m,y)` di tipo `date` è *valida* se le condizioni seguenti sono soddisfatte:

   - il mese m è compreso fra 1 e 12, dove 1 rappresenta gennaio, 2 febbraio, e così via;
   - il giorno `d` è compreso fra 1 e 31 per i mesi di gennaio, marzo, maggio, luglio, agosto, ottobre, dicembre;
   - il giorno `d` è compreso fra 1 e 30 per i mesi di aprile, giugno, settembre, novembre;
   - il giorno `d` è compreso fra 1 e 29 per il mese di febbraio negli anni bisestili, fra 1 e 28 per il mese di febbraio negli anni non bisestili.

   L'anno `y` può essere un intero qualsiasi. Un anno è bisestile se è multiplo di 4 ed inoltre o non è multiplo di 100 o è multiplo di 400.

   Suggerimento: Si usi la funzione `(mod) : int -> int -> int` che calcola il resto della divisione di due interi.

2. (5 punti) Si definisca una funzione `is_after : date -> date -> int` tale che `is_after dt1 dt2` sollevi una eccezione `NoComparison` precedentemente definita se `dt1` o `dt2` non è una data valida (nel senso del punto precedente), altrimenti restituisca:

   - 0 se `dt1` è uguale a `dt2`,
   - 1 se `dt1` è dopo `dt2`,
   - -1 se `dt1` è prima di `dt2`.

3. (10 punti) Usando la funzione `is_after` del Punto 2, si definisca una funzione `earliest : date list -> date option` tale che `earliest lst` restituisca `None` se la lista `lst` è vuota, e `Some dt` se `dt` la prima data in ordine di tempo nella lista non vuota `lst`. Che succede se una delle date in `lst` non è valida? Si giustifichi brevemente la risposta.