

**Programmazione Funzionale**  
**Prova scritta – Febbraio 2010**

**Nota bene:** la dichiarazione di qualunque funzione (anche locale) che non sia accompagnata dalla specifica del tipo e dalla descrizione dichiarativa di che cosa calcola la funzione non verrà presa in considerazione.

1. Scrivere una funzione

```
prendi: ('a -> bool) -> 'a list -> 'a * 'a list
```

che, applicata a un predicato  $p$  e una lista  $L$  sollevi un'eccezione se  $L$  non contiene alcun elemento che soddisfa  $p$ , altrimenti restituisca una coppia  $(x, L')$  dove  $x$  è un elemento di  $L$  che soddisfa  $p$  e  $L'$  contiene tutti gli altri elementi di  $L$ , in qualsiasi ordine. Se  $L$  contiene più occorrenze di  $x$ , dalla lista ne verrà rimossa soltanto una (in altri termini,  $L'$  ha solo un elemento in meno di  $L$ ).

Ad esempio, il valore di `prendi (function x -> x > 10) [3; 20; 7; 11; 8; 30; 20]` può essere  $(20, [3; 7; 11; 8; 30; 20])$ .

2. Si definisca un tipo di dati per la rappresentazione di alberi binari e scrivere un programma che, dato un albero  $A$  e una lista di elementi dello stesso tipo dei nodi di  $A$ , restituisca, se esiste, un ramo dell'albero dalla radice a una *foglia* che contenga tutti i nodi di  $L$  (in qualsiasi ordine) ed eventualmente anche altri nodi. Se un tale cammino non esiste, il programma solleverà un'eccezione. Si assuma che la lista  $L$  sia senza ripetizioni.
3. Si consideri la seguente dichiarazione di tipo per la rappresentazione di grafi orientati:

```
type 'a graph = ('a * 'a) list
```

Scrivere un programma che, dato un grafo  $G$ , un nodo  $N$  di  $G$  e una lista  $L$  di elementi dello stesso tipo di  $N$  restituisca, se esiste, un cammino che, partendo da  $N$ , contenga tutti i nodi di  $L$  (in qualsiasi ordine) ed eventualmente anche altri nodi. Se un tale cammino non esiste, il programma solleverà un'eccezione. Si assuma che la lista  $L$  sia senza ripetizioni.

Ad esempio, se  $G$  è il grafo rappresentato da  $[(1, 2); (1, 3); (1, 4); (2, 6); (3, 5); (4, 6); (6, 5); (6, 7); (5, 4)]$  e  $L$  è la lista  $[2; 5]$ , la ricerca a partire dal nodo 1 restituirà il cammino  $[1; 2; 6; 5]$ . Se la lista  $L$  è  $[2; 6; 4]$ , la ricerca a partire dal nodo 1 fallirà perché non esistono cammini in  $G$  che a partire da 1 tocchino tutti i nodi di  $L$ .

4. Si consideri la seguente dichiarazione:

```
let xxx y = List.for_all (function x -> List.mem x y)
```

Determinare il tipo di `xxx` e darne una specifica dichiarativa.