Esercizi proposti – 6

1. Definire:

   (a) `find: ('a -> bool) -> 'a list -> 'a`, tale che `find p lst` riporti il primo elemento di `lst` che soddisfa il predicato `p`. La funzione solleva un'eccezione se nessun elemento della lista soddisfa `p`. (Notare che il modulo `List` contiene una funzione con questo nome, ma qui si chiede di ridefinirla per esercizio).

   Utilizzare `find` per definire una funzione `find_applicata: int list -> int` che, applicata a una lista di interi, riporti il primo elemento della lista il cui quadrato sia minore di 30.

   (b) `takewhile: ('a -> bool) -> 'a list -> 'a list`, tale che `takewhile p lst` riporti la più lunga parte iniziale di `lst` costituita tutta da elementi che soddisfano il predicato `p`. Gli elementi del risultato devono occorrere nello stesso ordine in cui occorrono nell'argomento.

   Ad esempio, `takewhile (function n -> n mod 2 = 0) [0;2;4;6;7;8;9;10] = [0; 2; 4; 6]`.

   (c) `dropwhile: ('a -> bool) -> 'a list -> 'a list`, tale che `dropwhile p lst` riporti la lista che si ottiene eliminando i primi elementi di lst, fino a che soddisfano il predicato `p`. Gli elementi del risultato devono occorrere nello stesso ordine in cui occorrono nell'argomento.

   Ad esempio, `dropwhile (function n -> n mod 2 = 0) [0;2;4;6;7;8;9;10] = [7; 8; 9; 10]`.

   (d) `partition: ('a -> 'bool) -> 'a list -> ('a list * 'a list)`, tale che `partition p lst = (yes,no)`, dove `yes` contiene tutti gli elementi di lst che soddisfano il predicato `p`, e `no` quelli che non lo soddisfano. Gli elementi delle due liste `yes` e `no` possono essere in qualsiasi ordine.

   Ad esempio, il valore di `partition (function n -> n mod 2 = 0) [0;2;4;6;7;8;9;10]` può essere `([10; 8; 6; 4; 2; 0], [9; 7])`.

   (e) `pairwith: 'a -> 'b list -> ('a * 'b) list` tale che, `pairwith y [x1;x2;...;xn] = [(y,x1);(y,x2);....; (y,xn)]`. Utilizzare la funzione `List.map`.

   (f) `verifica_matrice: int -> int list list -> bool`, che, dato un intero n e una matrice di interi, rappresentata mediante liste di liste, riporta `true` se la matrice contiene almeno una riga i cui elementi siano tutti minori di n, `false` altrimenti. Utilizzare le funzioni `List.exists` e `List.for_all`.

   (g) `setdiff: 'a list -> 'a list -> 'a list`, la differenza insiemistica, utilizzando la funzione `List.filter`.

   (h) `subset: 'a list -> 'a list -> bool`, tale che `subset set1 set2 = true` se `set1` rappresenta un sottoinsieme di `set2`. Utilizzare la funzione `List.for_all`.

   (i) `duplica: int list -> int list`, che raddoppia tutti gli elementi di una lista di interi, usando la funzione `List.map`.

   Ad esempio, `duplica [0;1;2;3;4] = [0; 2; 4; 6; 8]`.

(j) `mapcons: ('a * 'b list) list -> 'b -> ('a * 'b list) list`, che che, data una lista di coppie `L` e un elemento `x: 'b`, riporta la lista che si ottiene inserendo `x` in testa a ogni secondo elemento delle coppie in `L`. Utilizzare la funzione `List.map`.

Ad esempio, applicata alla lista `[('A',[1;2]); ('B',[3;4;5]); ('C',[])]` e al valore 0, la funzione riporta `[('A',[0;1;2]); ('B',[0;3;4;5]); ('C',[0])]`.

(k) `tutte_liste_con: int -> 'a -> 'a -> 'a list list`, che, dato un intero non negativo `n` e due valori (dello stesso tipo) `x` e `y`, riporta una lista contenente tutte le possibili liste di lunghezza `n` contenenti soltanto i due valori `x` e `y`.

Ad esempio, per n=3, x=0 e y=1 si avra' la lista seguente (o una sua permutazione):

```
[[0; 0; 0]; [0; 0; 1]; [0; 1; 0]; [0; 1; 1];
 [1; 0; 0]; [1; 0; 1]; [1; 1; 0]; [1; 1; 1]]
```

(l) `interleave: 'a -> 'a list -> 'a list list`, tale che `interleave x lst` riporti una lista con tutte le liste che si ottengono inserendo `x` in qualsiasi posizione in `lst`. Utilizzare la funzione `List.map`.

Ad esempio, `interleave 10 [0;1;2] = [[10; 0; 1; 2]; [0; 10; 1; 2]; [0; 1; 10; 2]; [0; 1; 2; 10]]`.

(m) `permut: 'a list -> 'a list list`, tale che `permut lst` riporti una lista con tutte le permutazioni di lst (in qualsiasi ordine).

2. Sia data una matrice quadrata le cui caselle possono avere o no un contenuto, rappresentata da tre componenti:

   - la dimensione della matrice;
   - una lista associativa, che associa a ogni coppia di coordinate il suo contenuto se presente.

   Ad esempio, la variabile `labirinto` sotto dichiarata rappresenta una matrice del tipo considerato:

```
let labirinto = (5,
   [((1,0),"oro"); ((3,1),"oro"); ((4,3),"oro");
    ((0,1),"argento"); ((2,4),"argento"); ((0,2),"mostro");
    ((1,1),"mostro"); ((1,3),"mostro"); ((2,3),"mostro");
    ((3,0),"mostro"); ((4,2),"mostro")])
```

   Le caselle a cui non è associato alcun contenuto sono vuote. Una matrice è dunque rappresentata da un valore di tipo `int * ((int * int) * 'a) list`.

   Definire, usando `List.exists`, `List.for_all` e `List.find`:

   (a) una funzione `in_riga: (int * ((int * int) * 'a) list) -> int -> 'a -> bool`, che, data una matrice rappresentata come sopra, un numero di riga e un valore, verifichi se la riga data contiene il valore dato;

   (b) una funzione `trova_colonna: (int * ((int * int) * 'a) list) -> int -> 'a`, che, data una matrice rappresentata come sopra, un numero di riga `r` e un valore `v`, riporti il numero di colonna `c` tale che `(r,c)` contiene `v` (se esiste), altrimenti sollevi un'eccezione;

(c) una funzione `in_tutte: (int * ((int * int) * 'a) list) -> 'a -> bool`, che, data una matrice rappresentata come sopra e un valore, verifichi se tutte le righe della matrice contengono il valore dato.

3. (Esame di luglio 2011)

   (a) Scrivere una funzione `find: 'a -> 'a list -> 'a list * 'a list`, che, applicata a un elemento x e a una lista L, spezzi L in due parti: la prima contiene tutti gli elementi che vanno dall'inizio della lista fino alla prima occorrenza di x esclusa; la seconda contiene tutti gli elementi che seguono la prima occorrenza di x. La funzione solleverà un'eccezione se L non contiene x. Ad esempio, `find 3 [1;2;3;4;5;6;3] = ([1;2],[4;5;6;3])`.

   (b) Utilizzando la funzione `find` definita al punto precedente, definire una funzione `spezza: 'a -> 'a list -> 'a list * 'a list`, che, applicata a un elemento x e una lista L riporti una coppia di liste $(L1, L2)$, dove $L1$ contiene tutti gli elementi di L che vanno dalla prima alla seconda occorrenza di x, estremi esclusi, e $L2$ contiene tutti gli elementi di L che seguono la seconda occorrenza di x. La funzione solleverà un'eccezione se L non contiene almeno due occorrenze di x. Ad esempio, `spezza 3 [1;2;3;4;5;6;3;7;8;9;3;10] = ([4;5;6],[7;8;9;3;10])`.

4. (Dall'esame di Febbraio 2010) 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 elementi che soddisfano p, dalla lista ne verrà rimosso soltanto uno (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])`.