Programmazione Funzionale

Esercitazione 4 – Liste

Esercizio 1. Definire la funzione split di tipo `('a * 'b ) list -> ('a list * 'b list)` che associa ad ogni lista $[(a_1, b_1); \ldots; (a_n, b_n)]$ la coppia di liste $([a_1; \ldots; a_n], [b_1; \ldots; b_n])$.

Esercizio 2. Definire la funzione lgt che calcola la lunghezza di una lista.

Definire la funzione merge che data due liste $([a_1; \ldots; a_n], [b_1; \ldots; b_k])$ restituisce la lista $[(a_1, b_1); \ldots; (a_n, b_n)]$ quando $n = k$. Se invece $n \neq k$ la funzione sollevera un eccezione.

Esercizio 3. Definire la funzione invert di tipo $'a\ list \to' a\ list$ che inverta una lista.

Definire la funzione concat che prende due liste $[a_1; \ldots; a_n]$ e $[b_1; \ldots; b_k]$ e restituisce $[a_1; \ldots; a_n; b_1; \ldots; b_k]$.

Esercizio 4. Una lista di tipo `'a * 'b list` puo essere vista come una funzione *parziale* dal tipo `'a` verso il tipo `'b`.

Definire la funzione `comp : ('a * 'b) list * ('b * 'c) list -> ('a * 'c) list` che corrisponde alla composizione delle funzioni parziale. La funzione sollevera un eccezione se il dominio della prima e il codominio della secondo lista non coincidono.

Esercizio 5. Una *sottolista–prefisso* di una lista $[a_1; \ldots; a_n]$ a una lista della forma $[a_1; \ldots; a_k]$ dove $k \leq n$. Una *sottolista* di una lista $[a_1; \ldots; a_n]$ a una lista della forma $[a_i; \ldots; a_k]$ dove $1 \leq i, k \leq n$.

1. Definire una funzione prefix che data una lista l restituisce la lista di tutte le sottoliste prefisso di l.
2. Definire una funzione sublist che data una lista l restituisce la lista di tutte le sue sottoliste.

Esercizio 6. Una lista corrisponde a una sequenza finita $(a_1, \ldots, a_n)$ di elementi di un insieme A. Formalmente, $List(A) = \bigcup_{n \in \mathbb{N}} \{(a_1, \ldots, a_n) \mid a_i \in A\}$. La funzione contains e definita nel modo seguente:

$$\text{contains} : List(A) \times A \to \{true, false\}, ((a_1, \ldots, a_n), a) \mapsto \begin{cases} true \text{ if there exists } a_i = a. \\ false \text{ otherwise.} \end{cases}$$

1. Definire in OCaml la funzione contains per le liste di interi.
2. Per quale altri tipi α si puo definire contains per le $\alpha\ list$?
3. Definire la funzione witness che prende una lista di interi l e un intero n e restituisce una coppia di bool $\times$ int; $(true, i)$ se l contiene n dove i e la posizione nella lista di n. $(false, 0)$ se l non contiene n.

Esercizio 7. Un *occorrenza* in una lista $(a_1, \ldots, a_n)$ di un elemento a e un intero i tale che $a_i = a$.

1. Definire la funzione occurence che prende una lista di interi l e un intero n e restituisce una lista di interi che corrisponde a le occorrenze di n in l.
2. Definire la funzione remove che prende una lista di interi l e un intero n e toglie tutte le occorrenze n della lista l.
3. Definire la funzione contract che prende una lista di interi l e un intero n e toglie tutte le occorrenze di n meno la prima occorrenza di n della lista l.