PROGRAMMAZIONE I (A,B) - a.a. 2016-17
Terzo Appello – 15 Giugno 2017

Esercizio 1

Si scriva una funzione **C** che, dati due array a e b, entrambi di dimensione dim, verifica la seguente proprietà:

- ogni elemento contenuto nell'array a in una posizione con indice pari è presente anche nell'array b (in una posizione qualunque)
- ogni elemento contenuto nell'array a in una posizione con indice dispari è minore di tutti gli elementi contenuti in b

La funzione deve restituire 1 se la proprietà è verificata, 0 altrimenti.

SOLUZIONE E' conveniente dare una soluzione modulare, prevedendo funzioni ausiliarie che veficano, rispettivamente, se un valore è presente in un array e se è minore di tutti i valori di un array.

```
int presente(int b[], int dim, int x) {
  int i=0;
  int trovato=0;
  while (i<dim && !trovato) {
    if (b[i] == x) trovato=1;
    i++;
  }
  return trovato;
}

int minore(int b[], int dim, int x) {
  int i=0;
  int ok=1;
  while (i<dim && ok) {
    if (x>=b[i]) ok=0;
    i++;
  }
  return ok;
}

int check(int a[], int b[], int dim) {
  int i=0;
  int ok=1;
  while (i<dim && ok) {
    if ((i%2==0 && !presente(b,dim,a[i])) || (i%2==1 && !minore(b,dim,a[i])))
      ok=0;
    i++;
  }
  return ok;
}
```

Esercizio 2

Dato il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c\}$

$$\mathcal{L} = \{ca^n cb^m c \mid n < 2m\}$$

si verifichi se il linguaggio è regolare o meno (fornendo una opportuna dimostrazione) e si definisca una grammatica che lo genera.

SOLUZIONE Il legame fra gli esponenti di a e di b dovrebbe far sospettare che il linguaggio non sia regolare. Si noti innanzitutto che il vincolo $n < 2m$ implica che $m > 0$, quindi in ogni parola del linguaggio è presente almeno un b

Per dimostrarlo applicando il *pumping lemma*, dato un qualunque numero naturale n, prendiamo la stringa $w = ca^n cb^n c$. Le possibili decomposizioni $w = xyz$ sono:

- $x = \epsilon$, $y = ca^t$ e $z = an - tcb^n c$, con $0 \leq t \leq n - 1$
- $x = ca^s$, $y = a^t$ e $z = a^{n-s-t} cb^n c$, con $0 \leq s \leq n - 2$ e $1 \leq t \leq n - 1 - s$

Nel primo caso abbiamo che $xy^0 z \notin \mathcal{L}$ poiché tale stringa non contiene il simbolo c iniziale. Nel secondo caso, $xy^{2n} z \notin \mathcal{L}$ poiché tale stringa contiene almeno $2n$ occorrenze di a ed esattamente n occorrenze di b.

Una grammatica che genera il linguaggio è la seguente:

```
S -> cTbc | caTbc
T -> c | AATb
A -> a | _
```

dove `_` rappresenta la stringa vuota.

Esercizio 3

Si suppongano predefiniti i tipi

```
struct el {int info; struct el *next;};
typedef struct el ElementoDiLista;
typedef ElementoDiLista* ListaDiElementi;
```

Si scriva in **C** una procedura che, presa una lista e un intero x, ridispone gli elementi della lista in ordine inverso rimuovendo al contempo tutti gli elementi che contengono il valore x.

SOLUZIONE

```
void invertirimuovi (ListaDiElementi *l, int x) {
  ListaDiElementi l2 = NULL;
  while (*l != NULL) {
    ListaDiElementi tmp = *l;
    *l = (*l)->next;
    if (tmp->info == x) {
      free(tmp);
    }
    else {
      tmp->next = l2;
      l2 = tmp;
    }
  }
  *l = l2;
}
```

Esercizio 4

Si definisca in CAML, senza usare la ricorsione esplicita, una funzione

`sommadopozero : int list -> int`

che, data una lista `lis` di interi, calcola la somma di tutti gli elementi che seguono la prima occorrenza del valore 0 nella lista. Se la lista non contiene nessuno 0, il risultato calcolato dalla funzione deve essere 0. Definire la funzione senza rovesciare la lista.

SOLUZIONE

```
let sommadopozero lis =
  let f x (y1,y2) =
    if (x!=0) then (y1+x,y2)
              else (0,y1+y2)
  in
    let (a,b) = foldr f (0,0) lis in b
```