PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2014-2015
Prova scritta del 12/01/2015

SOLUZIONI PROPOSTE

Per ogni esercizio vengono proposte una o più soluzioni tra le tante possibili.

ESERCIZIO 1 (6 punti)
Dato l'alfabeto $\Lambda = \{a, b, \$\}$ si definisca una grammatica libera che genera il seguente linguaggio:

$$\mathcal{L} = \{\$a^n\$b^n\$ \mid n > 0\}$$

Si dimostri, utilizzando il pumping lemma, che tale linguaggio non è regolare.

Soluzione

```
S ::= $A$
A ::= aAb | a$b
```

Per dimostrare che il linguaggio non è regolare utilizzando il pumping lemma mostriamo quanto segue:

$$\forall n \in \mathbb{N}.\ \exists w \in \mathcal{L}.\ \mid w \mid \geq n \wedge \forall x, y, z. \neg\big(w = xyz\ \wedge\ y \neq \epsilon \wedge\ \mid xy \mid \leq n\ \wedge\ (\forall i \in \mathbb{N}. xy^i z \in \mathcal{L})\big)$$

ovvero

$$\forall n \in \mathbb{N}.\ \exists w \in \mathcal{L}.\ \mid w \mid \geq n \wedge \forall x, y, z. \big(w = xyz\ \wedge\ y \neq \epsilon \wedge\ \mid xy \mid \leq n) \Rightarrow\ \neg(\forall i \in \mathbb{N}. xy^i z \in \mathcal{L})\big)$$

Sia allora $n \in \mathbb{N}$ e sia

$$w = \$a^n\$b^n\$$$

Ovviamente $\mid w \mid \geq n$. Siano dunque x, y, z tali che:

(a) $w = xyz$

(b) $y \neq \epsilon$

(c) $\mid xy \mid \leq n$

Ragioniamo per casi:
(1) $x = \epsilon\ \wedge\ y = \$a^k$, con $0 \leq k \leq n-1$ (e in questo caso $z = a^{(n-k)}\$b^n\$$).
La stringa $xy^0z = z = a^{(n-k)}\$b^n\$ \notin \mathcal{L}$ c.v.d.

(2) $x = \$a^k\ \wedge\ y = a^h$, con $h > 0$ e $h + k \leq n - 1$ (e in questo caso $z = a^{(n-h-k)}\$b^n\$$).
La stringa $xy^0z = xz = \$a^k\$b^n\$ \notin \mathcal{L}$ c.v.d.

ESERCIZIO 2 (6 punti)
Si definisca in CAML, senza utilizzare ricorsione esplicita, una funzione

```
insord : 'a -> 'a list -> 'a list
```

in modo che `(insord x lis)`, supponendo `lis` ordinata in modo non decrescente, restituisca la lista ottenuta inserendo `x` in `lis` e ordinata nello stesso modo. Ad esempio:

```
insord 4 [-1; 0; 3; 3; 6; 10] = [-1; 0; 3; 3; 4; 6; 10]
```

```
insord 'b' ['d'; 'g'; 'h'] = ['b'; 'd'; 'g'; 'h']
```

Soluzione

```
let insord x l =
   let f z y = match y with
       w::ws when w<=z -> w::z::ws |
       w::ws when w > z -> z::w::ws
   in
foldr f [x] l;;
```

Una seconda soluzione siottiene utilizzando la funzione di ordine superiore `filter` come segue:

```
let insord x l =
   let
      minugx z = z<=x
   and
      magx z = z>x
   in
      (filter minugx l) @ [x] @ (filter magx l)
```

ESERCIZIO 3 (6 punti)
Si scriva in `C` una procedura che, data una lista di interi in cui tutti gli elementi con valore pari precedono tutti gli elementi con valore dispari, e dato un intero `n`, inserisce `n` nella lista in modo che anche nella lista risultante tutti gli elementi con valore pari precedano tutti gli elementi con valore dispari.
Si suppongano predefiniti i tipi

```
struct el {int info; struct el *next;};

typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiInteri;
```

Soluzione
Inseriamo l'elemento in testa alla lista se l'elemento è pari oppure la lista è vuota. Inseriamo l'elemento in fondo alla lista altrimenti.

```
void ins (ListaDiInteri *l, int n)
{
   ListaDiInteri new = malloc (sizeof (ElementoDiLista));
   new -> info = n;
   if (n % 2 == 0 || *l==NULL)
   {
      new->next = *l;
      *l = new;
   }
   else
   {
     ListaDiElementi prec=NULL, corr=*l;
     while (corr != NULL)
        {
          prec = corr;
          corr = corr->next;
        }
     prec->next = new;
     new->next = NULL;
}
```

ESERCIZIO 4 (6 punti)
Si scriva una funzione `C`

```
int check (int a[], int dim)
```

che, dato un array `a` e la sua dimensione `dim`, restituisce il valore di verità della seguente formula

$$\big(\forall i \in [0, dim-1).\ (\exists j \in [i+1, dim).\ a[i] > a[j])\big)$$

Soluzione
La proprietà è soddisfatta solo se il valore minimo dell'array occorre una sola volta e in ultima posizione.

```
int check (int a[], int dim)
{
   int posmin=0, int occmin=1;
   int j;
   for (j=1; j<dim;j++)
       if (a[j]==a[posmin]) occmin=occmin+1;
       else
          if (a[j]<a[posmin]) {posmin=j; occmin=1;}
   return (posmin=dim-1 && occmin=1);
}
```

ESERCIZIO 5 (6 punti)
Scrivere in `CAML` una funzione

```
multiset : 'a list -> ('a * int) list
```

che, data una lista `lis` di elementi, restituisce una lista di coppie in cui ciascuna coppia contiene un elemento di `lis` e il numero di volte in cui tale elemento occorre in `lis`. Nella lista risultante tutti i primi elementi delle coppie devono essere distinti tra loro.
Ad esempio:

```
multiset [20;40;30;20;50;20;30] = [(20,3); (40,1); (30,2); (50,1)]
```

Suggerimento: si utilizzino definizioni di funzioni ausiliarie.

Soluzione
Definiamo una funzione ausiliaria che inserisce un elemento in un multiinsieme e la utilizziamo per inserire ricorsivamente gli elementi della lista nel multinsieme risultato.

```
let rec multiset lis =
   let rec ins el m = match m with
       [] -> [(el,1)] |
       (x,n) :: ms when x=el -> (x,n+1)::ms |
       (x,n)::ms when x<>el -> (x,n)::(ins el ms)
   in
      match lis with
        [] -> [] |
        x::xs -> ins x (multiset xs);;
```