PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2013/14

Prova scritta del 3 giugno 2014

Scrivere **IN STAMPATELLO** COGNOME, NOME, MATRICOLA e CORSO su ogni foglio consegnato

ESERCIZIO 1
Dato l'alfabeto $\Lambda = \{a, b, c\}$, si definisca una grammatica che genera il seguente linguaggio

$$\mathcal{L} = \{a^n \alpha b^n \mid n > 0 \wedge (pari(n) \Rightarrow \alpha \in \{b, c\}^*) \wedge (dispari(n) \Rightarrow \alpha \in \{c\}^+)\}$$

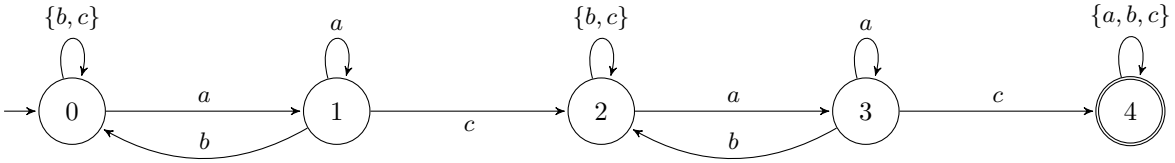
dove *pari* e *dispari* hanno l'ovvio significato.

Soluzione

```
S ::= aAb
A ::= D | aPb | aaAbb | ab
P ::= b | c | bP |cP
D ::= c | cD
```

ESERCIZIO 2
Definire un automa **deterministico** che riconosce le stringhe sull'alfabeto $\Lambda = \{a, b, c\}$
che contengono almeno due volte la sottostringa ac.

Soluzione



ESERCIZIO 3
Dato il tipo degli alberi binari

```
type 'a btree = Void  | Node of 'a * 'a btree * 'a btree
```

si definisca in CAML una funzione foo con tipo

```
   foo : 'a btree -> int * int
```

in modo che (foo bt) sia la coppia costituita dal numero di nodi foglia con etichetta maggiore della radice e dal numero di nodi foglia con etichetta minore o uguale della radice (la funzione non è definita su alberi vuoti).

Soluzione

```
let foo bt =
    let rec conta bt x = match bt with
        Void -> (0,0)   |
        Node(y, Void, Void) -> if y>x then (1,0) else (0,1)
        Node (_, lbt, rbt) when lbt<>Void or rbt <>Void ->
                let (al, bl) = conta lbt x
                and (ar, br) = conta rbt x
                in  (al+ar, bl+br)
    in
       match bt with
         Node(y,lbt,rbt) -> conta bt y;;
```

ESERCIZIO 4
Si definisca in `C` una funzione

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

che restituisce il valore di verità della seguente formula:

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

Soluzione

```
int check (int a[], int dim)
{
   int trovato, j, i, ok;
   j=1; trovato=0;
   while (j<dim-1 && !trovato)
   {
     i=0;
     ok = 1;
     while (i<j && ok)
        if (a[i]<=a[j]) ok=0; else i++;
     if (!ok)
     {
       i=j+1;
       ok=1;
       while (i<dim && ok)
        if (a[i]>=a[j]) ok=0; else i++;
     }
    trovato=ok; j++;
   }
   return trovato;
}
```

ESERCIZIO 5
Senza utilizzare ricorsione esplicita, definire in CAML una funzione

```
foo : 'a list -> ('a -> bool) -> bool
```

in modo che `(foo lis p)` restituisca `true` se il numero di elementi di `lis` che soddisfano `p` è maggiore del numero di elementi di `lis` che non soddisfano `p`; restituisca `false` altrimenti.

Soluzione

```
let foo lis p = let f x y = if (p x) then y+1 else y-1 in (foldr f 0 lis)>0;;
```

ESERCIZIO 6
Date le seguenti definizioni:

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

scrivere in C una procedura che, dati in ingresso attraverso opportuni parametri una lista di interi e un intero x, elimina dalla lista l'ultimo elemento maggiore di x. Se nessun elemento della lista è maggiore di x, la procedura elimina, se esiste, l'ultimo elemento della lista.

Soluzione

```
void canc (ListaDiELementi *l, int x)
{
 if (*l != NULL)
 {
   ListaDiElementi corr=*l, prec=NULL, prec_prec=NULL, prec_ultimo=NULL;
   int trovato=0;

   while (corr != NULL)
      {
         if (corr->info>x) {trovato=1; prec_ultimo=prec;}
         prec_prec=prec;
         prec=corr;
         corr=corr->next;
      }

   if (trovato)
      if (prec_ultimo==NULL) {corr= *l; *l=*l->next; free(corr);}
      else
         {corr=prec_ultimo->next; prec_ultimo->next=corr->next; free(corr);}
   else
      if (prec_prec==NULL) {corr= *l; *l=*l->next; free(corr);}
      else {corr=prec_prec->next; prec_prec->next=NULL; free(corr);}

  }
 }
```