PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2012/13

Prova scritta del 6 giugno 2013

SOLUZIONI PROPOSTE

ESERCIZIO 1 (punti 5)
Si definisca una grammatica libera che genera il linguaggio

$$\mathcal{L} = \Lambda^* \setminus \{a^n b^n \mid n \geq 0\}$$

sull'alfabeto $\Lambda = \{a, b\}$.

Soluzione
Il linguaggio richiesto puó essere caratterizzato nel modo seguente:

$$\{a^n b^m \mid (n+m) > 0 \wedge n \neq m\} \cup \{b\alpha \mid \alpha \in \Lambda^*\} \cup \{a\alpha b\beta a\gamma \mid \alpha, \beta, \gamma \in \Lambda^*\}$$

```
S ::= L | M | N
L ::= aLb | A | B
A ::= aA | a
B ::= bB | b
M ::= bP
N ::= aPbPaP
P ::= ϵ | aP | bP
```

ESERCIZIO 2 (punti 5)
Si considerino le seguenti definizioni di tipo:

```
type 'a unodue = Uno of 'a  | Due of 'a * 'a
type 'a mseq = Single of 'a unodue | Mult of 'a unodue * 'a mseq
```

Si definisca una funzione count con tipo

```
count : 'a mseq -> int
```

in modo che (count m) restituisca il numero di elementi di tipo 'a presenti in m.

Soluzione

```
let rec count =
     let count_unodue ud = match ud with
          Uno _ -> 1  |
          Due(_,_) -> 2
     in
     match m with
        Single ud -> countud ud  |
        Mult(ud, m1) -> (countud ud) + (count m1);;
```

ESERCIZIO 3 (punti 5)
Dato un array a di dimensione dim sia:

$$mscart(a) = \texttt{max}\{|(a[i] - a[j]| \ | \ i, j \in [0, dim) \ \ \wedge i \neq j\}$$

Si definisca in C una funzione
boolean check (int a[], int dim, int d)
che, dati un array a, la sua dimensione dim e un intero d, restituisce il valore di verità $mscart(a) < d$.

(**N.B.** $|x - y|$ indica la differenza in valore assoluto tra x e y.
Si assuma predefinito il tipo typedef enum {false, true} boolean).

Soluzione

```
int abs (int x, int y)
   {
     int val_ass;
     val_ass = x-y;
     if (val_ass < 0) val_ass = - val_ass;
     return (val_ass);
   }


boolean check (int a[], int dim, int d)
{
  int mscart = 0;
  int aux, i, j;
  for (i=0; i<dim-1; i++)
    for (j=i+1; j<dim; j++)
       {
          aux = val_ass(a[i], a[j]);
          if (aux > mscart) mscart = aux;
       }
  return (mscart < d);
}
```

ESERCIZIO 4 (punti 5)
Si completi la seguente definizione

```
let foo p lis = let f x y = ....  in foldr f ... lis
```

della funzione

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

in modo che (foo p lis) restituisca:

- il primo elemento di lis che soddisfa p, se tale elemento esiste
- il primo elemento di lis, altrimenti

```
let foo p lis = let f x y = if p x then x else y  in match lis with z::zs -> foldr f z lis;;
```

ESERCIZIO 5 (punti 5)
Date le seguenti definizioni:

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

scrivere in C una procedura che, data in ingresso una lista di interi, inserisce in testa alla lista un nuovo elemento che contiene la somma di tutti gli interi della lista.

Soluzione

```
void ins_sum (ListaDiElementi *lis)
  {
   ListaDiElementi nuovo = malloc(sizeof(ElementoDiLista));
   int somma = 0;
   int scorri = *lis;
   while (scorri != NULL)
      {
         somma = somma + scorri -> info;
         scorri = scorri -> next;
      }
   nuovo -> info = somma;
   nuovo -> next = *lis;
   *lis = nuovo;
  }
```

ESERCIZIO 6 (punti 5)
Si definisca una procedura C che, dato un array di interi a di dimensione dim, sostituisce ogni elemento con la somma tra l'elemento stesso e tutti gli elementi che lo precedono in a.

Soluzione

```
void sost (int a[], int dim)
    {
      int i;
      for (i=1; i<dim; i++)
        a[i] = a[i] + a[i-1];
    }
```