PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2011-2012
Prova scritta del 5/09/2012

SOLUZIONI PROPOSTE

Negli esercizi di programmazione in `C` si assuma predefinito il tipo `typedef enum {false, true} boolean`.

ESERCIZIO 1 (punti 6)
Una *lingua* è una struttura dati che contiene valori e che può terminare sia con uno che con due valori. Nel primo caso si parla di lingua *schietta*, nel secondo di lingua *biforcuta*. Il tipo polimorfo `CAML` per rappresentare le lingue è dato da

```
type 'a tongue = Tongue of 'a * 'a tongue | Frank of 'a | Bifurcate of 'a*'a;;
```

Si scriva una funzione

```
purge : 'a tongue list -> 'a tongue list
```

che elimina da una lista di lingue tutte quelle biforcute.

Soluzione

```
let rec purge tl =
     let rec isfrank t = match t with
         Frank _ -> true |
         Bifurcate _,_ -> false |
         Tongue _, t1 -> isfrank t1

     in

     [] -> [] |
     t::ts  -> if (isfrank t) then t :: (purge ts) else (purge ts);;
```

ESERCIZIO 2 (punti 6)
Si definisca in `C` una funzione

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

che restituisce il valore di verità della seguente formula:

$$(\forall i \in [3, dim-3).\ \#\{j \mid j \in [i-3, i+3) \wedge a[i] = a[j]\} < 2)$$

Soluzione

```
boolean check (int a[], int dim)
{
  int i, j, conta;
  boolean trovato = false;
  i=3;
  while (i<dim-3 && !trovato)
  {
    conta=0;
    for (j=i-3; j<i+3; j++)
      if (a[i]==a[j]) conta=conta+1;
    if (conta < 2) i=i+1; else trovato=true;
  }
  return (!trovato);
}
```

ESERCIZIO 3 (punti 6)
Con riferimento all'esercizio 1, si dia una grammatica che genera tutti e soli i valori del tipo `bool tongue`, indicando esplicitamente, oltre alle produzioni, l'alfabeto, le categorie sintattiche e il simbolo iniziale.

Soluzione
Definiamo $G = \langle \Lambda, V, S, P \rangle$ con

- $\Lambda = \{$ `,` , Tongue , Frank , Bifurcate , true , false , (,)$\}$
- $V = \{$S,B$\}$
- $S =$ S
- Le produzioni in P sono le seguenti:
  S ::= Tongue(B`,` S) | Frank B | Bifurcate(B`,`B)
  B ::= true | false

ESERCIZIO 4 (punti 6)
Si completi la seguente definizione

```
let sum l = let f x y = ... in foldr f ... l
```

in modo che

```
sum : int tongue list -> int list
```

e `(sum tlist)` restituisca una lista di interi in cui ciascun valore è uguale alla somma di tutti i valori nella *lingua di interi* che si trovava nella stessa posizione nella lista passata come argomento.

Soluzione

```
let sum tlist = let f x y =
                   let rec sumt t = match t with
                      Frank(n) -> n |
                      Bifurcate(n,m) -> n+m |
                      Tongue(n, t1) -> n + (sumt t1)
                   in
                     (sumt x)::y
     in
        foldr f [] tlist;;
```

ESERCIZIO 5 (punti 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 due interi x e y e una lista di interi, inserisce x e y immediatamente prima del primo valore positivo nella lista. Se la lista non contiene valori positivi, x e y devono essere inseriti in coda alla lista.

Soluzione

```
void ins (int x, int y, ListaDiElementi *lis)
{
       ListaDiElementi before=NULL, p=*lis, newx, newy;
       boolean trovato = false;

       newx=malloc(sizeof(ElementoDiLista));
       newy=malloc(sizeof(ElementoDiLista));
       newx -> info = x;
       newy -> info = y;
       newx -> next = newy;

       while (!trovato && p!=NULL)
            if (p->info > 0)
              trovato = true:
            else
              {  before = p;
                 p=p->next;
              }
       newy -> next = p;

       if (before==NULL)
          *lis = newx;
       else
          before -> next = newx;
}
```