PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2011-2012
II Verifica scritta del 19/12/2011

SOLUZIONI PROPOSTE

ESERCIZIO 1 (6 punti)
Si consideri il seguente tipo di alberi *ternari* in cui ogni nodo ha al più tre figli.

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

Si definisca in `CAML` una funzione `check_bin` con tipo

```
check_bin : 'a ttree -> bool
```

in modo che `check_bin alb` restituisca *true* se `alb` è anche un albero binario, e restituisca *false* altrimenti.

Soluzione

```
let rec check_bin tt = match tt with
     Void -> true |
     Node(_, lt, mt, rt) when lt <> Void && mt <> Void && rt <> Void -> false |
     Node(_, lt, mt, rt) when lt = Void or mt = Void or rt = Void  ->
                                check_bin lt && check_bin mt && check_bin rt;;
```

ESERCIZIO 2 (6 punti)
Scrivere in `C` una funzione

```
void foo (int a [], int b [], int dima, int dimb)
```

che, dati due array `a` e `b` di dimensione `dima` e `dimb` rispettivamente, rimpiazza ogni elemento di `a` con il numero di sue occorrenze in `b`. Dati ad esempio i seguenti array `vet1` e `vet2`

3	1	5

5	2	5	3	7	9

la chiamata `foo(vet1, vet2, 3, 6)` deve modificare `vet1` come segue

1	0	2

Soluzione

```
void foo (int a[], int b[], int dima, int dimb)
  {
    int ia, ib, conta;
    for (ia=0; ia < dima; ia++)
      {
         conta = 0;
         for (ib=0; ib < dimb; ib++)
           if (a[ia]==b[ib]) conta=conta+1;
         a[ia]=conta;
      }
  }
```

ESERCIZIO 3 (6 punti)
Scrivere in C una procedura

```
void azzera (int a[], int dim)
```

che azzera tutti gli elementi dell'array compresi tra il primo elemento pari ed il secondo elemento pari (l'array rimane inalterato se non contiene almeno due elementi pari ovvero se i primi due elementi pari sono consecutivi).

Soluzione

```
void azzera (int a[], int dim)
  {
    int i=0, ind_primo, ind_secondo, n_pari=0;
    while (i<dim && n_pari < 2)
       {
         if (a[i] % 2 == 0)
              {
                 if (n_pari==0) ind_primo=i; else ind_secondo=i;
                 n_pari = n_pari + 1;
              }
         i=i+1;
        }
    if (n_pari==2)
      for (i=ind_primo+1; i<ind_secondo; i++)
          a[i]=0;
  }
```

ESERCIZIO 4 (6 punti)
Date le seguenti definizioni:

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

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

scrivere in C una procedura che, data in ingresso attraverso un opportuno parametro una lista di interi, elimina tutti i multipli del valore massimo in essa contenuta (compreso il valore massimo medesimo).

Soluzione

```
void del_mul_max (ListaDiElementi *lista)
   {
      ListaDiElementi aux = *lista;
      int val_max;
      boolean elimina_primo;

      if (aux != NULL)
        {
          val_max = aux -> info;
          aux = aux -> next;
          while (aux != NULL)
             {
                if ((aux -> info) > val_max)
                    val_max = aux->info;
                aux = aux -> next;
             }

          if ((*lista) -> info % val_max == 0)
                     elimina_primo=true;
          else       elimina_primo = false;

          while (*lista != NULL && elimina_primo)
             {
               if ((*lista) -> info % val_max == 0)
                 {
                    aux = *lista;
                    *lista = (*lista) -> next;
                    free(aux);
                 }
               else  elimina_primo = false;
             }
          if (*lista != NULL)
             {  ListaDiElementi prec = *lista;
                aux = prec -> next;
                while (aux != NULL)
                   if (aux -> info % val_max == 0)
                  {    prec -> next = aux -> next;
                       free(aux);
                       aux = prec -> next;   }
               else  { prec = prec -> next;
                       aux = aux -> next;   }
             }
          }
      }
```

ESERCIZIO 5 (6 punti)
Si supponga di estendere la sintassi dei comandi con la seguente produzione:

```
Com ::= reset Ide;
```

la cui semantica (rispetto al modello senza memoria dinamica) è data dalla seguente regola:

$$\mathcal{S}em_c \texttt{ reset id } \rho\ \mu = \mu[^{?}/_{\rho(\texttt{id})}]^{mod}$$

Estendere opportunamente il tipo `Com` e la definizione della funzione `semc` nella implementazione CAML della semantica dei comandi, al fine di contemplare anche il nuovo comando.

Soluzione
Estensione del tipo `Com`

```
type com = ... |
           Reset of ide;;
```

Estensione di `semc`

```
let rec semc c (a:amb list) (m:mem list) = match c with
     ...
     Reset(x) -> let (Def l) = search a x
                 in  update_stack m l (Def Unknown)
```