Programmazione I e Laboratorio

SOLUZIONE - II verifica intermedia del 20/12/2016

1) Date le definizioni

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

si definisca una procedura C che prende due liste e cancella dalla prima gli elementi che compaiono esattamente una volta nella seconda.

POSSIBILE SOLUZIONE:

```
int conta(ListaDiElementi l, int x)
{
  int cont=0;
  while (l!=NULL)
  {
     if (l->info==x) cont++;
     l = l->next;
  }
  return cont;
}

void cancella(ListaDiElementi* l1, ListaDiElementi l2)
{
  ListaDiElementi prec = NULL;
  ListaDiElementi curr = *l1;
  while (curr!=NULL)
  {
    if (conta(l2,curr->info)==1)
    {
      if (prec==NULL)
      {
        *l1 = *l1->next;
        free(curr);
        curr = *l1;
      }
      else
      {
         prec->next = curr->next;
        free(curr);
        curr = prec->next;
      }
    }
    else
    {
      prec = curr;
      curr = curr->next;
    }
  }
}
```

2) Si definisca inoltre una procedura C che, presa una lista ordinata in modo non crescente e un intero `n`, inserisce `n` nella lista in modo che continui ad essere ordinata in modo non crescente.

POSSIBILE SOLUZIONE:

```
int inserisci(ListaDiElementi *l, int n)
{
  ListaDiElementi prec = NULL;
  ListaDiElementi curr = *l;
  ListaDiElementi new = malloc(sizeof(ElementoDiLista));
  int trovato = 0;

  new->info = n;

  while (curr!=NULL && !trovato)
  {
    if (curr->info <= n) trovato=true;
    else
    {
      prec = curr;
      curr = curr->next;
    }
  }
  if (prec==NULL)
  {
    new->next = *l;
    *l = new;
  }
  else
  {
    new->next = curr;
    prec->next = new;
  }
}
```

3) Definire una funzione ricorsiva CAML

`canc: 'a list -> 'a -> 'a list`

Tale che (`canc lis n`) cancella da `lis` l'ultima occorrenza del valore `n`. Se `n` non compare la lista viene restituita immutata.

POSSIBILE SOLUZIONE:

```
let rec canc lis n =
  let rec member l m =
    match l with
        [] -> false
      | x::xs -> x=m or member xs m
  in
  match lis with
      [] -> []
    | x::xs -> if (x=n && !member xs n) then xs
                                        else x::canc xs n;;
```

4) Definire la funzione `canc` dell'esercizio precedente, senza utilizzare ricorsione esplicita. Suggerimento: è bene ricordare, durante il calcolo del risultato, se l'elemento è stato cancellato o no.

POSSIBILE SOLUZIONE:

```
let canc lis n =
  let f x y =
    match y with
        (xs,false) -> if (x=n) then (xs,true)
                                else (x::xs,false)
      | (xs,true) -> (x::xs,true)
  in
    let (l,c) =  foldr f ([],false) lis
  in
    l;;


// oppure, al posto delle ultime tre righe:
// fst (foldr f ([],false) lis);;
```