PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2011/12

Prova scritta del 25 giugno 2012

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

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

ESERCIZIO 1 (punti 6)
Il tipo `Cicoria di 'a` è definito come segue:

- La cicoria vuota ha tipo `Cicoria di 'a`
- Una foglia di cicoria (rappresentata da una lista di `'a`) e una `Cicoria di 'a` hanno tipo `Cicoria di 'a`

Il tipo CAML e' dato da

```
type 'a chicory = EmptyChic | Chic of 'a list * 'a chicory;;
```

Si scriva una funzione CAML che prende una cicoria di interi `c` e un intero `x` e restituisce true se `x` compare in tutte le foglie di cicoria di `c`.

ESERCIZIO 2 (punti 6)
Sia definito il predicato *be* nel modo seguente
$be(a, b, j, dim) \equiv j \in [0, dim) \ \wedge \ (\forall i \in [0, j).\ a[i] = b[dim - i - 1])$
Si scriva una funzione `C`

```
int be_index(int a[], int b[], int dim)
```

dove `a` e `b` sono array di interi di dimensione `dim` che restituisce

- il valore -1 se $\nexists j.\ j \in [0, dim) \wedge be(a, b, j, dim)$
- il massimo valore dell'insieme
  $\{i \mid i \in [0, dim) \ \wedge \ be(a, b, i, dim)\}$
  altrimenti

Suggerimento: si cerchi di comprendere il significato della proprietà espressa dal predicato *be*.

ESERCIZIO 3 (punti 6)
Scrivere una funzione `C`

```
int subseq (int a[], int b[], int dima, int dimb)
```

dove `dima` e `dimb` sono le dimensioni degli array `a` e `b` rispettivamente, che calcoli il numero di volte in cui l'array `a` compare come sottosequenza dell'array `b`.

ESERCIZIO 4 (punti 6)
Si completi la seguente definizione

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

con tipo

```
split  : 'a list -> 'a list * 'a list
```

in modo che `(split lista)` restituisca la coppia `(l1, l2)` tale che `l1@l2=lista`, gli elementi di `l2` siano tutti uguali tra loro e l'ultimo elemento di `l1`, se esiste, è diverso dagli elementi in `l2`

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 un intero `x` e una lista di interi, sposta in ultima posizione il primo elemento della lista, se esiste, strettamente maggiore di `x`.