PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2008-2009
Prova scritta del 11 febbraio 2009

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

ESERCIZIO 1 (punti 6)
Si definisca una grammatica libera che genera il seguente linguaggio sull'alfabeto $\Lambda = \{a, b\}$

$$L = \{\alpha \mid \alpha \in \Lambda^+, |\alpha_a| = |\alpha_b|\}$$

dove $|\alpha_x|$ indica il numero di occorrenze del simbolo x nella stringa α.

ESERCIZIO 2 (punti 6)
Si definisca la funzione di ordine superiore `mapif`. La chiamata `mapif p f l` restituisce la lista ottenuta da `l` applicando la funzione `f` a ogni elemento sul quale il predicato `p` è vero, e lasciando inalterati gli elementi sui quali il predicato `p` è falso. Si dia anche il tipo di `mapif`.

ESERCIZIO 3 (punti 6)
Si definisca in `C` una procedura

```
void med (int  vet[], int dim, int *ris)
```

che, supponendo `dim` dispari e gli elementi del vettore `vet` (di dimensione `dim`) tutti distinti tra loro, restituisce in `*ris` l'indice del valore mediano di `vet`, ovvero dell'elemento che è maggiore di $\lfloor$`dim/2`$\rfloor$ elementi in `vet` e minore di $\lfloor$`dim/2`$\rfloor$ elementi in `vet` (si ricorda che $\lfloor x \rfloor$ indica la parte intera di un numero reale x).
Ad esempio, se `a` è l'array seguente

12	1	15	3	22	33	7	44	25

la chiamata `med(a, 9, pos)` deve calcolare in `*pos` il valore `2`, essendo 15 il valore mediano.

ESERCIZIO 4 (punti 6)
Si supponga di estendere la sintassi delle dichiarazioni con la nuova produzione

```
Dec ::= Type *Ide pointsto Ide
```

Informalmente, `T *x pointsto y` ha l'effetto di dichiarare la variabile `x` facendo in modo che il suo valore sia un puntatore alla variabile `y`, che si suppone essere già presente nello stato. Dare la semantica della nuova dichiarazione, in riferimento al modello semantico **completo** in cui lo stato è costituito da ambiente, memoria e heap.

ESERCIZIO 5 (punti 6)
Senza utilizzare ricorsione esplicita, ma utilizzando funzioni di ordine superiore, si definisca una funzione `sumsplit` con tipo

```
sumsplit: int list -> int -> (int * int)
```

in modo che, se `(sumsplit lista x) = (s1, s2)`, `s1` è la somma di tutti e soli gli elementi di `lista` minori o uguali a `x` e `s2` è la somma di tutti e soli gli elementi di `lista` maggiori di `x`.