PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2008-2009
Prova scritta del 10 luglio 2009

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

ESERCIZIO 1 (punti 6)
Data la seguente grammatica libera sull'alfabeto $\Lambda = \{a, b\}$

$$S \longrightarrow aaSb \mid ab$$

dimostrare formalmente (senza fare uso di alberi di derivazione) che la stringa *aaaaabbb* appartiene al linguaggio generato dalla grammatica.

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

```
void SeqStartEnd (int  vet[], int dim, int *start, int *end)
```

che, dato un array `vet` di dimensione `dim`, lascia nelle variabili puntate da `start` e `end` i valori degli indici `i,j` tali che la porzione del vettore compresa tra `i` e `j` inclusi è una delle sottosequenze non descrescenti in `vet` di lunghezza massima.

Ad esempio, dato l'array `v` in figura

12	1	15	33	3	2	44	88	6	18

la chiamata `SeqStartEnd(v, 10, &inizio, &fine)` deve lasciare nelle variabili `inizio` e `fine` i valori `1` e `3` (oppure i valori `5` e `7`) rispettivamente.

ESERCIZIO 3 (punti 6)
Si supponga di estendere la sintassi delle dichiarazioni con la nuova produzione

```
Dec ::= T *Ide ? instack Ide:new
```

L'effetto della dichiarazione `T *p ? instack q:new`, dove si assume che `q` sia una variabile di tipo `T *` già presente nello stato, è quello di dichiarare la variabile `p` di tipo `T *`, attribuendole:

- il valore del puntatore `q` se quest'ultimo punta ad una variabile sullo stack;
- il puntatore ad un nuovo elemento creato nello heap altrimenti.

ESERCIZIO 4 (punti 6)
Si definisca una funzione `check` con tipo

```
check: ('a -> 'b -> bool) -> 'a list -> 'b list -> bool
```

in modo che `(check r lista1 lista2)` restituisca `true` se e soltanto se esiste un elemento `x` in `lista1` tale che vale `(r x y)` per ogni `y` in `lista2`.

ESERCIZIO 5 (punti 6)
Dato il tipo degli alberi binari visto a lezione

```
type 'a btree = Empty  | Node of 'a * 'a btree * 'a btree
```

si definisca una funzione `addleft : 'a btree -> 'a -> 'a btree -> 'a btree` in modo che `(addleft bt x lt)` sia l'albero ottenuto aggiungendo ad ogni nodo etichettato con `x` e il cui sottoalbero sinistro è vuoto, un sottoalbero sinistro uguale a `lt`, come nel seguente esempio.