PROGRAMMAZIONE II mod A – gr. 1
a.a. 2016/2017
prof. Giuliano Laccetti

PROVA D'ESAME del 14.06.2017

ESERCIZIO 0.
sbarramento (obbligatorio per tutti: chi non fa bene questo esercizio non è ammesso all'orale, indipendentemente dal risultato degli altri esercizi)

a) Dati 2 stack `head1` e `head2` "ordinati", progettare in P-like un algoritmo per la costruzione di uno stack `head3`, anch'esso "ordinato", merge tra `head1` e `head2`.
Ad esempio. Se

```
head1  = 2-> 7-> 8   ; head2 =  1-> 3-> 4-> 5-> 6
head3 sarà  1-> 2-> 3 -> 4-> 5-> 6-> 7-> 8
```

(per inserire/estrarre/visualizzare elementi, utilizzare **ESCLUSIVAMENTE** le *procedures* `pop(head, elem)` e `push(head, elem)` che, rispettivamente, estraggono, eliminandolo, un elemento dallo stack e lo restituiscono in `elem`, e inseriscono l'elemento `elem` nello stack).

b) Dato un **BST**, ordinato secondo il campo `key`, progettare in P-like un algoritmo, sotto forma di *function ricorsiva* (`function ricerca_nodo_in_BST_ric(root, key)`) per la ricerca di un nodo, con campo `key` dato, nel **BST** . L'output consisterà nel puntatore al nodo che contiene la chiave `key`, `NULL` altrimenti.

ESERCIZIO 1.
Data una linked list `head` con il campo `info` di tipo intero progettare un algoritmo ricorsivo in P-like che elimini dalla lista tutti gli elementi che nel campo `info` contengono il valore, dato, `ELEM`.
Si organizzi l'algoritmo in 2 procedure, entrambe ricorsive, una che elimina `ELEM` dalla testa, `del_testa(head, ELEM)` , una che elimina `ELEM` dal mezzo della lista, `del_mezzo(head,ELEM)` , ed una terza procedura, `delete_elem (head, ELEM)` , che utilizza le altre due.

ESERCIZIO 2.

a)
Progettare in P-like un algoritmo ***ricorsivo*** sotto forma di function (`function costrList_ric`) di tipo puntatore a nodo_lista, che costruisca una linked list con i campi info di tipo intero, contenenti i numeri da 1 a N, in ordine crescente. **Indicare anche un esempio di chiamata da un main con i valori dei parametri attuali.**

b)
stesso esercizio, versione iterativa. `function costrList_it`

ESERCIZIO 3.
Progettare in P-like un algoritmo ricorsivo sotto forma di function (`logical function precedente_maggiore_sommasuccessivi`) che data una linked list `head` (con campo info di tipo `integer`), restituisca `TRUE` se ciascun valore nei campi info è maggiore della somma di tutti i valori dei campi info degli elementi successivi, `FALSE` altrimenti.
Utilizzare una function ricorsiva, **pure da progettare**, in P-like, (`integer function somma_valori_campinfosuccessivi`) , che, data una linked list, restituisca la somma dei valori dei campi info (di tipo `integer`) dei suoi elementi