Programmazione I - Laboratorio - Esercizi con le liste

Usare la seguente definizione negli esercizi seguenti:

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

Esercizio 1 Scrivere una funzione stampaLista che ricevuta una `ListaDiElementi`, la stampi a video in questo modo: `1 -> 2 -> 3 -> 4 -> //`

Esercizio 2 Definire una funzione (fornirne due versioni, una iterativa e una ricorsiva) `lunghezzaLista` che data una `ListaDiElementi`, restituisca la sua lunghezza

Esercizio 3 Definire una funzione deallocaLista che riceve una `ListaDiElementi` e ne dealloca tutti gli elementi.

Esercizio 4 Definire una funzione (fornirne due versioni, una iterativa e una ricorsiva) `primoPari` che data una `ListaDiElementi`, restituisca il puntatore al primo elemento pari nella lista (restituisce `NULL` se la lista e' vuota o non contiene elementi pari)

Esercizio 5 Definire una funzione (fornirne due versioni, una iterativa e una ricorsiva) `minimoPari` che data una `ListaDiInteri`, restituisca il puntatore al minimo elemento pari nella lista (restituisce `NULL` se la lista e' vuota o non contiene elementi pari)

Esercizio 6 Definire una procedura (sia iterativa che ricorsiva) 'elimina' che ricevuta una `ListaDiElementi` e un intero `X`, elimini (senza deallocare) i primi `X` elementi e ritorni il puntatore alla testa della lista modificata

Esercizio 7 Definire una funzione ordinaLista che modifica una `ListaDiElementi` data ordinandola in modo crescente. La funzione non deve usare allocazione dinamica della memoria (`malloc` e `free`), ne modificare il campo `info` degli elementi. La funzione restituisce il puntatore al primo elemento ottenuto dopo l'ordinamento

Esercizio 8 Definire una funzione `merge` che date due `ListaDiElementi` ordinate, restituisca una nuova `ListaDiElementi` ordinata contenente tutti gli elementi delle due liste. Le liste originali devono restare immutate