

Esercitazione 9

Nella propria home directory creare una sottodirectory chiamata es9, in cui metteremo tutti i file C di oggi.

Quando dovete usare o ritornare dei valori booleani, usate la seguente definizione:

```
typedef enum {false, true} boolean;
```

Inoltre quando un esercizio vi chiede di scrivere un "predicato", dovete scrivere una funzione che ritorni un valore di tipo boolean.

Ricordate che oltre alla funzione o procedura indicata dall'esercizio, si richiede che scriviate anche un main che usi tale funzione e ne dimostri *ogni* funzionalità.

In alcuni degli esercizi che seguono (quando richiesto) si utilizzi la seguente definizione di tipo "lista di interi":

```
typedef struct El {  
int info;  
struct El *next;  
} ElementoListaInt;  
typedef ElementoListaInt* ListaDiInteri;
```

Inoltre, dove c'è bisogno di allocazione dinamica della memoria, dovete assicurarvi di deallocare SEMPRE tutta la memoria allocata.

Ricordate infine che dovete includere la libreria stdlib.h per la gestione della memoria dinamica.

Esercizi introduttivi:

1) Scrivere una funzione stampaLista che ricevuta una ListaDiInteri, la stampi a video in questo modo:

```
1 -> 2 -> 3 -> 4 -> / /
```

Usare questa funzione per verificare le funzioni dei prossimi esercizi.

2) Scrivere una procedura che, data una lista di interi, restituisca l'i-esimo elemento della lista stessa.

3) Definire una funzione (fornirne due versioni, una iterativa e una ricorsiva) lunghezzaLista che data una ListaDiInteri, restituisca la sua lunghezza.

4) Si consideri una lista di interi, definire una funzione che inserisce un valore intero in coda ad una lista.

- 5) Si consideri una lista di interi. Scrivere una funzione `trovaIntero`, che dato un intero restituisce `true` se l'elemento è presente all'interno della lista. `False` in tutti gli altri casi (lista vuota, elemento non presente). La procedura termina immediatamente se l'intero è presente nella lista.
- 6) Scrivere una funzione (una versione iterativa ed una ricorsiva) in cui dato un valore intero `x` ed una lista di interi, si conta le occorrenze di `x` nella lista.

Esercizi:

- 7) Definire una procedura `stampaInverso`, che data una lista di interi la stampi in maniera invertita.
- 8) Definire una funzione `primoPari` che data una lista di interi, restituisca il primo elemento pari nella lista, oppure `NULL` in caso di lista vuota o senza elementi pari.
- 9) Definire una procedura (sia iterativa che ricorsiva) `'elimina'` che ricevuta una `ListaDiInteri` e un intero `X`, elimini i primi `X` elementi e restituisca la lista modificata.
- 10) Definire una funzione (sia iterativa che ricorsiva) `minimoPari` che data una `ListaDiInteri`, restituisca il minimo elemento pari nella lista (restituisce `NULL` se la lista è vuota o non contiene elementi pari).
- 11) Definire una procedura (sia iterativa che ricorsiva) `inserisciQuarto` che data una `ListaDiInteri` ed un intero `P`, inserisca `P` dopo il terzo elemento di lista. Quest'ultima viene lasciata inalterata se non contiene almeno tre elementi.
- 12) Data una lista di interi, scrivere una procedura che crei una nuova lista contenente solo i valori che nella prima lista compaiono due volte. La prima lista deve rimanere intatta.
- 13) Definire una funzione `ordinaLista` che modifica una `ListaDiInteri` data ordinandola in modo crescente. La funzione non deve usare allocazione dinamica della memoria (`malloc` e `free`). Suggerimento: la testa può cambiare? Quindi come deve essere passata la lista alla funzione?

14) Definire una procedura 'merge' che date due ListaDiInteri ordinate, restituisca una nuova ListaDiInteri ordinata contenente tutti gli elementi delle due liste. Le liste originali devono restare immutate