ISTRUZIONI GENERALI

Spedire in UN'UNICA MAIL, da inviare agli indirizzi

finocchi@di.uniroma1.it e spognardi@di.uniroma1.it

un file .c contenente esclusivamente le implementazioni delle funzioni descritte di seguito (ed eventuali funzioni ausiliarie necessarie per la corretta compilazione ed esecuzione). Non includere il main e non inviare alcun file con estensione ".h".

Non correggeremo esercizi inviati separatamente ai due indirizzi o ad uno solo di essi. Non prenderemo inoltre in considerazione esercizi anonimi, esercizi inviati alla Prof.ssa Fachini, o esercizi spediti fuori dal tempo massimo per la consegna.

Potete svolgere ogni esercizio implementando ricorsivamente la funzione di cui viene dato il prototipo, oppure richiamando da tale funzione una funzione ausiliaria (purche' implementata ricorsivamente) con eventuali parametri che possano risultare utili nelle chiamate ricorsive.

I tipi per la gestione delle liste da utilizzare sono quelli usati durante le ore di esercitazione.

Data una lista L, indicheremo per brevita' con L[i] il suo i-esimo elemento, dove i e' un valore >=1 e <= lunghezza di L.

ESERCIZIO 1
Data una lista di interi, modificarla eliminando gli elementi in posizione pari che sono multipli dell'elemento nella posizione precedente, assumendo che le posizioni siano numerate a partire da 1. In altre parole, eliminare dalla lista gli elementi L[2k] che sono multipli di L[2k-1], dove k e' un valore intero >=1 e <=(lunghezza di L)/2. Il prototipo della funzione da implementare e' il seguente:

```
void eliminaMultipli(ListPtr L)
```

Il fatto che il tipo del parametro sia ListPtr e non ListPtr * non e' un errore.

Ad esempio, se la lista contiene gli elementi

2 6 7 9 1 4 3 2 4

la lista modificata deve contenere

2 7 9 1 3 2 4

ESERCIZIO 2
Diremo che una lista e' k-alternata se i primi k+1 elementi formano una sequenza strettamente crescente, gli elementi dal (k+1)-esimo al (2k+1)-esimo formano una sequenza strettamente decrescente, gli elementi dal (2k+1)-esimo al (3k+1)-esimo formano una sequenza strettamente crescente, e via dicendo. L'ultima sequenza deve essere necessariamente decrescente e puo' contenere anche meno di k+1 elementi.

Ad esempio, la lista 1 3 8 9 2 1 0 3 4 5 4 2
                       < < < > > > < < < > >
e' 3-alternata.

Implementare una funzione che, dati una lista ed un intero k, restituisca 1 se la lista e' k-alternata, e 0 altrimenti. Il prototipo della funzione da implementare e' il seguente:

```
int kalternata(ListPtr L, int k)
```

ESERCIZIO 3
Scrivere una funzione che, date due liste L1 ed L2 della stessa lunghezza, modifichi L1 in base alla seguente regola:

- se L2[i]<0, rimuove da L1 l'elemento L1[i];

- se L2[i]>0, aggiunge, agganciandola a L1[i], la lista con L2[i] elementi che contengono nell'ordine i valori L1[i]+1, ..., L1[i]+ L2[i]. Quest'ultimo elemento deve essere agganciato a L1[i+1].

Ad esempio, se le due liste sono

L1 15 70 40 2 10

L2 -1 3 2 0 -5

allora L1 deve essere modificata come segue:

L1 70 71 72 73 40 41 42 2

Il prototipo della funzione da implementare e' il seguente:

```
void modifica(ListPtr *L1, ListPtr L2)
```