Esercitazione 10

Argomento: Tipi di dato astratti

Per questa esercitazione si consiglia di organizzare il codice come segue:

- fare un file .h dove mettere le intestazioni delle funzioni
- fare un file .c contenente l'implementazione delle funzioni
- scrivere il main in un altro file .c in cui si testano le funzioni implementate

Code

Esercizio 10.1

Implementare una versione con side effect senza condivisione di memoria del tipo di dati astratto **Coda**, rappresentato tramite una SCL, che contenga I puntatori al primo e all'ultimo elemento della coda.

```
typedef int T ;

struct NodoSCL {

T info ;

struct NodoSCL * next ;

};

typedef struct NodoSCL TipoNodo ;


struct CodaColl {

TipoNodo * head;

TipoNodo * tail;

} ;

typedef struct CodaColl CodaColl;

typedef CodaColl * Coda;
```

Implementare perciò le seguenti funzioni:

- Coda * codaVuota ()
- bool estVuota (Coda * c)

- void inCoda (Coda * c , T e)
- void outCoda (Coda * c)
- T primo (Coda * c)
- T ultimo (Coda * c)

Ogni funzione va implementata con costo O(1).

Liste

Si consideri il tipo di dati astratti lista rappresentato mediante una SCL.

```
typedef int T ;

struct NodoSCL {

T info ;

struct NodoSCL * next ;

};

typedef struct NodoSCL TipoNodo ;

typedef TipoNodo * TipoLista ;
```

Vi vengono forniti I file lista.c e lista.h, contenenti l' implementazione del tipo secondo uno schema funzionale con condivisione di memoria, e le funzioni che avete visto a lezione su questo tipo di dato (*length*, *append*, *concat*, *ins, cons* e *get*).

Utilizzare le funzioni fornite per svolgere gli esercizi seguenti senza accedere direttamente alla SCL che rappresenta la lista:

Esercizio 10.2

Implementare la funzione C

T sommaElementi(TipoLista l);

che data in ingresso una lista la restituisca la somma dei suoi elementi, oppure 0 se la lista è vuota.

Esercizio 10.3

Implementare la funzione C

TipoLista halfMerge(TipoLista l1, TipoLista l2);

che date in ingresso due liste l1 e l2 restituisca una nuova lista formata dalla primà metà della prima lista concatenata con la seconda metà della seconda lista.

Es: input: 2→3→7→18→2→9 4→29→3→17

output: 2→3→7→3→17

Esercizio 10.4

Implementare la funzione ricorsiva C

TipoLista appendNtimes (TipoLista l , T e, int n);

che data in ingresso una lista l, un elemento e un intero positivo n, restituisca una lista a cui è stato aggiunto n volte l'elemento e in fondo alla lista.