Laboratorio di Programmazione 1

Liste

LISTE (1)

Sia data una lista i cui nodi sono definiti tramite la seguente struttura C:

```
struct nodo{
        int valore;
        struct nodo *next;
};
```

Scrivere due funzioni C RICORSIVE che ricevendo in ingresso un puntatore alla lista: (1) stampano la lista e (2) contano quanti multipli di 3 sono presenti nella lista.

NB: per testare il programma scrivere la funzione inserisci in testa per creare una lista con 10 valori.

LISTE (2)

Si consideri una lista dinamica di interi, i cui elementi sono del tipo definito come di seguito riportato:

```
typedef struct El {
        int dato1;
        int dato2;
        struct El *next; } ELEMENTO;
```

Si codifichi in C le funzioni **contamultipli** e **multiplo**. La funzione **contamultipli** riceve come parametro la testa della lista e restituisce il numero di elementi della lista che hanno il primo valore numerico (dato1) multiplo del secondo (dato2). Se la lista è vuota, la funzione restituisce il valore -1.

Continua …

Liste (2)

~~Per verificare se un valore intero è multiplo di un altro la funzione deve richiamare la funzione ricorsiva **multiplo** definita dallo studente.~~

ESERCIZIO DA TEMA D'ESAME (

Si completi il file (skeleton) ESI_20062018_A_1.c di modo che:

- la funzione **`crea_lista`** riceve in input una stringa, converte la stringa in una lista di caratteri e restituisce il puntatore alla testa della lista. La struttura `struct node_t` deve essere utilizzata come definizione dei nodi della lista.
- la funzione **`stampa_lista`** stampa la lista in input.
- la funzione **`raddoppia_vocali`** riceve in input una lista (il puntatore al primo nodo) e raddoppia i nodi della lista che contengono delle vocali.

L'output del programma é il seguente:

```
la mia stringa
laa miiaa striingaa
```

NOTA: Lo studente NON deve gestire la deallocazione della lista.

LISTE (2)

- Completare il programma list.c rispettando le specifiche fornite nei commenti del codice stesso. In particolare e richiesto di implementare:
- *1) int is_empty(struct list *L)* – la funzione deve restituire 1 se e solo se *L* è vuota, 0 altrimenti;
- 2) *void list_print(struct list *L)* – la funzione deve stampare a video il contenuto della lista *L* (tenere presente che *L* puo anche essere la lista vuota, ossia valere NULL);
- 3) *struct list *list_search(struct list *L, int k)* – la funzione deve restituire un puntatore ad un nodo della lista *L* in cui il campo *val* contenga il valore *k*; se il valore *k* non compare nella lista, la funzione deve restituire NULL;
- 4) *struct list *nth_element(struct list *L, int n)* – la funzione deve restituire il puntatore all'*n*-esimo nodo della lista (*n*=0 è il primo nodo, *n*=1 è il secondo nodo, ecc.). Se la lista ha meno di *n*+1 nodi, deve restituire NULL;
- 5) *struct list *list_from_array(int v[], int n)* – la funzione deve creare una nuova lista contenente gli *n* elementi dell'array *v*;
- 6) *struct list *list_concat(struct list *L1, struct list *L2)* – la funzione deve restituire una lista contenente i nodi di L1 seguiti da quelli di L2. N.B., la funzione non deve creare nuovi nodi, ma utilizzare quelli gia presenti nelle due liste, "aggiustando" eventualmente alcuni puntatori. Come per altre delle funzioni precedenti, è possibile una semplice implementazione ricorsiva (vedi lucidi).
- 7) *struct list *list_reverse(struct list *L)* – la funzione deve restituire la lista ottenuta "rovesciando" i nodi di *L* (l'ultimo nodo di *L* diventa il primo della nuova lista; il penultimo diventa il secondo, e così via). Questa funzione non deve creare nuovi nodi, deve semplicemente riorganizzare quelli di *L*. Per questa funzione conviene probabilmente un approccio iterativo (non ricorsivo), in cui si esegue un ciclo il cui corpo consiste nello spostamento del primo nodo di *L* all'inizio della lista rovesciata. Il ciclo termina quando *L* diventa la lista vuota...

Da tema d'esame globale

- Si modifichi il file ESA 15062017 B 2.c in modo tale che:
  - la funzione encode ritorni una stringa in base ai valori delle matrici delle lettere e delle loro relative occorrenze
  - la funzione print_ma_mi stampi le matrici delle lettere e delle loro relative occorrenze
  - la funzione get_size ritorni il numero di occorrenze totali delle lettere
  - l'output della funzione main sia il seguente:

afk
oup
wej

302
201
032

Codifica: aaakkoopeeejj

DA TEMA D'ESAME

Si completi il file (skeleton) ESI_20062018_A_2.c di modo che:

- **`new_cell`** crea una nuova cella con nome e valore specificato
- **`init_matrix`** crea una matrice della dimensione specificata. Le celle assumono per righe i nomi 'A', 'B',...e il valore specificato
- **`print_matrix`** stampa la matrice specificata coi nomi delle celle se il corrispondente valore è diverso da 0 altrimenti stampa un BLANK.
- **`swap_cells`** scambia i nomi e i valori delle celle di nome name1 e name2. Se tali celle non sono presenti nella matrice specificata non fa nulla.

Se tutto è corretto, l'esecuzione del programma fornirà il seguente output:

```
A B C
D E F
G H I
F B C
D E A
G H I
```

NOTA: Lo studente NON deve gestire la deallocazione della lista.

DA TEMA D'ESAME GLOBALE

- Si modifichi il file ESA 15062017 B 1.c tale che:
  - la funzione **leggi** legga da tastiera una stringa lunga al più 256 caratteri
  - la funzione **crea_lista** crei una lista di caratteri a partire da una stringa s escludendo i caratteri numerici. Tale funzione deve essere ricorsiva.
  - la funzione stampa stampi la lista di caratteri in input ponendo una freccietta tra gli elementi della lista (si veda esempio di seguito).
  - Un esempio di esecuzione del main già implementato nel file è:

    Inserire una stringa: c0a09se3t367t8a

    Creazione della lista...

    La lista è: c -> a -> s -> e -> t -> t -> a