Esercitazione 9

Nella propria home directory creare una sottodirectory chiamata es09, in cui metteremo tutti i file C di oggi.

Esercizi strutture:

Attraverso le strutture potete mettere insieme più cose di tipo diverso.
Esempio:

```
struct anagrafe {
     char[100] nome;
     char[100] cognome;
     int eta;
     int numeroFigli;
     char sesso;
};
```

Come dichiaro un elemento di questa struttura? allo stesso modo dei tipi primitivi.
Per int avevamo: `int x`, oppure `int x[N]` per i vettori
Ora abbiamo:

```
struct anagrafe x, struct anagrafe x[N]
```

Le uniche operazioni valide sulle strutture sono:

- assegnare una struttura ad un'altra struttura dello stesso tipo
- rilevare l'indirizzo (&) di una variabile di tipo struttura
- accedere ai membri di una variabile di tipo struttura
- usare l'operatore sizeof per determinare la dimensione di una struttura

N.B. le strutture non si possono confrontare con == e != (Vanno confrontati i singoli elementi che compongono la struttura)
Le strutture possono essere passate alle funzioni:

- Passando l'intera struttura
- Passando singoli membri

In entrambi i modi il passaggio è per valore.
Per passare le strutture per riferimento, passare l'indirizzo.
Un vettore di strutture è passato per riferimento (è un vettore!). Creando una struttura contenente un vettore e passandola ad una funzione si ottiene il passaggio di un vettore per valore (se non si passa il riferimento della struttura e si modifica il vettore all'uscita della funzione il vettore della struttura rimane lo stesso).

1) Scrivere una funzione che dato un vettore di interi restituisca qual è il valore minimo, qual è il valore massimo e qual è la media dei valori del vettore. Utilizzare la seguente struttura:

```
typedef struct mat {
int min;
int max;
double media;
} mat;
```

2) Scrivere un nuovo tipo di dato coppia di interi.
Inizializzare tre istanze di coppie con i primi tre numeri naturali e i loro doppi.

3) Allocare dinamicamente tre istanze del tipo di dato coppia appena definito, inizializzarle con i valori dei primi tre numeri primi e dei loro quadrati e quindi dellaocarle.

4) Scrivere una funzione che riceva un array di coppie di interi e le inizializzi con i primi n numeri dispari (n e' la lunghezza del vettore) e i loro quadrati.

5) Scrivere una funzione che dato un vettore di coppie restituisca qual è la somma di tutti i valori del vettore.

Esercizi su ordinamenti e ricerca

La ricerca binaria e' un metodo per cercare un elemento in un array ordinato con un numero di confronti al massimo log(N) dove N e' il numero di elementi dell'array.

L'algoritmo e' simile al metodo usato per trovare una parola sul dizionario:
sapendo che il vocabolario e' ordinato alfabeticamente, l'idea e' quella di iniziare la ricerca non dal primo elemento, ma da quello centrale, cioe' a meta' del dizionario. Si confronta questo elemento con quello cercato:
- se corrisponde, la ricerca termina indicando che l'elemento e' stato trovato;
- se e' inferiore, la ricerca viene ripetuta sugli elementi precedenti (ovvero sulla prima meta' del dizionario), ignorando quelli successivi;
- se invece e' superiore, la ricerca viene ripetuta sugli elementi successivi (ovvero sulla seconda meta' del dizionario), ignorando quelli precedenti.

Quando la dimensione del sotto-array raggiunge 0, la ricerca termina indicando che il valore non e' stato trovato.

6) Scrivere una funzione ricorsiva che riceva un array di interi e lo ordini, usando l'algoritmo di Merge Sort.

7) Scrivere una funzione (sia iterativa che ricorsiva) che riceva un array ordinato, chieda all'utente un valore e ricerchi il valore nell'array usando l'algoritmo di ricerca binaria spiegato in alto. Usare una variabile globale per contare il numero di chiamate alla funzione effettuate e verificare la logaritmicita' con il numero di elementi dell'array.

Esercizi sulle liste

Negli esercizi che seguono si utilizzi la seguente definizione di tipo "lista di interi":

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

Per verificare che il programma 'esempio' deallochi tutta la memoria, usare il comando valgrind, che invoca il programma passato e controlla che la memoria sia tutta libera all'uscita:

```
valgrind ./esempio
```

Ricordate che dovete includere la libreria `stdlib.h` per la gestione della memoria dinamica.
Per controllare se quanto avete fatto è corretto, usate e modificate se necessario, la procedura stampaLista qui sotto:

```
/*=========================================================*/
     void stampaLista ( ListaDiInteri l ){
          if (l == NULL ) printf (" //\n" );
          else
          {
             printf ( "%d -> " ,l->info );
             stampaLista (l->next );
          }
     }
/*=========================================================*/
```

8) Scrivere un programma che crei una lista di 3 interi e li inizializzi ai primi tre naturali.

9) Scrivere un programma che crei dinamicamente una lista di 3 interi e li inizializzi ai primi tre naturali.
Il programma deve deallocare correttamente la lista prima di uscire, verificare con valgrind che questo sia avvenuto (se correttamente installato sul vostro pc).

10) Scrivere un programma che crei dinamicamente una lista di 3 interi e li inizializzi con valori chiesti all'utente.
Il programma deve deallocare correttamente la lista prima di uscire, verificare con valgrind che questo sia avvenuto (se correttamente installato sul vostro pc).

11) Scrivere un programma che chieda all'utente un numero N e crei una lista con N elementi, inizializzata con i primi N naturali
Il programma deve deallocare correttamente la lista prima di uscire, verificare con valgrind che questo sia avvenuto (se correttamente installato sul vostro pc).