Note

È considerato errore qualsiasi output non richiesto dagli esercizi.
È importante scrivere il proprio main in Visual Studio per poter fare correttamente il debug delle funzioni realizzate!

Esercizio 1 (5 punti)

Nel file `conta.c` implementare la definizione della seguente funzione:

```
extern unsigned int contanumeri(const char* nomefile);
```

La funzione accetta come parametro un nome di file (stringa C zero terminata) da aprire in modalità lettura tradotta (testo). La funzione deve contare quante cifre numeriche sono presenti nel file e ritornarne la quantità. Con "cifre numeriche" si intendono i byte corrispondenti ai simboli 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Ad esempio, utilizzando il file `file1.txt`:
`abc123ABC`
La funzione ritornerà 3.

Esercizio 2 (punti 6)

Creare i file `matrix.h` e `matrix.c` che consentano di utilizzare la seguente struttura:

```
struct matrix {
    size_t rows, cols;
    double *data;
};
```

e la funzione:

```
extern struct matrix *mat_rendiquadrata(const struct matrix *a);
```

La struct consente di rappresentare matrici di dimensioni arbitraria, dove `rows` è il numero di righe, `cols` è il numero di colonne e `data` è un puntatore a `rows×cols` valori di tipo `double` memorizzati per righe. Consideriamo ad esempio la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

questo corrisponderebbe ad una variabile `struct matrix A`, con `A.rows = 2`, `A.cols = 3` e `A.data` che punta ad un area di memoria contenente i valori `{ 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 }`.

La funzione accetta come parametro un puntatore a una matrice e deve ritornare una matrice allocata dinamicamente sull'heap ottenuta rendendo quadrata la matrice `a`, aggiungendo, se necessario, righe o colonne di zeri in basso o a destra. Il puntatore alla matrice non sarà mai `NULL`.

Ad esempio, utilizzando la matrice A precedente, `mat_rendiquadrata(A)` ritorna la matrice:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Esercizio 3 (7 punti)

Nel file `serpentino.c` implementare la definizione della seguente funzione:

```
extern char *serpentino_encode(const char *s);
```

La funzione accetta una stringa C (zero terminata) e ritorna una stringa C allocata dinamicamente sull'heap, contenente la stringa originale codificata con l'Alfabeto Serpentino. L'alfabeto serpentino è un gioco linguistico per bambini che consiste nel raddoppiare ogni vocale con l'aggiunta di una "s" interposta: per esempio, "a" diventa "asa", "e" diventa "ese", e così via (quindi "ciao" diventa "cisiasaoso").

Il puntatore ritornato deve puntare ad un'area di memoria allocata dinamicamente sull'heap grande esattamente il numero di byte necessari a contenere la stringa risultante e il terminatore e se `s` è NULL deve ritornare NULL.

Esercizio 4 (7 punti)

Creare i file `rapporti.h` e `rapporti.c` che consentano di utilizzare la seguente struttura:

```
#include <stdint.h> // Necessario per il tipo uint16_t

struct rapporto {
    uint16_t id;
    char tipo;
    float valore;
};
```

e la funzione

```
extern struct rapporto *leggi_rapporti(const char *filename, uint16_t *n);
```

È stato definito un formato binario di dati per memorizzare rapporti trasmessi da sensori su diverse macchine. Un rapporto è costituito da un campo `id` (intero senza segno a 16 bit codificato in little endian), seguito da un campo tipo (carattere alfabetico) e da un valore (numero in virgola mobile a 32 bit codificato in little endian). I rapporti sono salvati su un file che inizia con un intero senza segno a 16 bit codificato in little endian che indica il numero di rapporti presenti nel file, seguito poi da tanti rapporti come indicato.
Ad esempio il file `rapporto1.bin` contiene i seguenti byte (rappresentati in esadecimale nel seguito):

```
03 00 FF 00 78 7B 14 5E 40 1B 00 74 66 66 F7 42 04 00 6B 00 00 80 3F
```

Questo file contiene 3 rapporti (come indicato dai primi 16 bit `03 00`):

- `FF 00  78  7B 14 5E 40` id = 255, tipo = 'x', valore = 3.47
- `1B 00  74  66 66 F7 42` id = 27, tipo = 't', valore = 123.7
- `04 00  6B  00 00 80 3F` id = 4, tipo = 'k', valore = 1.0

La funzione `leggi_rapporti`, deve aprire in modalità lettura non tradotta (binario) il file il cui nome viene fornito dalla stringa C `filename` e caricarne il contenuto in memoria. La funzione deve

- Impostare la variabile puntata da `n` al numero di rapporti presenti sul file
- Allocare dinamicamente sull'heap la memoria necessaria a contenere tutti i rapporti.
- Leggere tutti i rapporti nell'area di memoria allocata.

La funzione deve ritornare il puntatore all'area di memoria allocata sull'heap se è riuscita ad aprire il file e a leggerne interamente il contenuto, NULL altrimenti.

Esercizio 5 (8 punti)

L'Imposta sui Redditi delle Persone Fisiche (IRPEF) viene calcolata come percentuale dello stipendio (lordo), suddivisa a scaglioni:

- la parte da 0 fino a 15000 euro è tassata al 23%,
- la parte oltre i 15000 e fino a 28000 è tassata al 27%,
- la parte oltre i 28000 e fino a 55000 è tassata al 38%,
- la parte oltre i 55000 e fino a 75000 è tassata al 41%,
- la parte oltre i 75000 è tassata al 43%.

Se lo stipendio è di 25000 euro, i primi 15000 sono tassati al 23%, la parte da 15001 a 25000 è tassata al 27%. In totale si calcola 15000*23/100 + (25000-15000)*27/100 = 6150.

Se lo stipendio è di 35000 euro, i primi 15000 sono tassati al 23%, la parte da 15001 a 28000 è tassata al 27%, la parte da 28001 a 35000 è tassata al 38%. In totale si calcola 15000*23/100 + (28000-15000)*27/100 + (35000-28000)*38/100 = 9620.

Nel file `tasse.c` implementare la definizione della seguente funzione:

```
extern unsigned int irpef( unsigned int stipendio, unsigned int *scaglioni,
                           unsigned int *aliquote, size_t n);
```

Nel vettore `scaglioni` sono presenti le `n` soglie a partire da `0`, nel vettore `aliquota` sono presenti le `n` percentuali. Nel caso illustrato (quello delle tasse italiane ad oggi) `scaglioni` punta a `{ 0, 15000, 28000, 55000, 75000 }`, `aliquote` punta a `{ 23, 27, 38, 41, 43 }`. La funzione deve calcolare le tasse secondo il procedimento illustrato per qualsiasi valore di stipendio e per qualsiasi coppia di vettori `scaglioni` e `aliquote`.