LEGGETE LA GUIDA PER LA CREAZIONE DEI PROGETTI E PER IL DEBUGGING!

Gli esercizi seguenti devono essere risolti, compilati e testati utilizzando il debugger. ***Per ognuno si deve realizzare un file main.c*** *che contenga la funzione main() che ne testi il funzionamento. Fate progetti diversi per ogni esercizio.*

Esercizio 1

Nel file leggiscrivi.c implementare le definizioni delle funzioni:

```
extern int txt2bin(const char *srcTxtFile, const char *dstBinFile);

extern int bin2txt(const char *srcBinFile, const char *dstTxtFile);
```

Entrambe le funzioni accettano come parametri due nomi di file, sotto forma di stringhe C.
Per la funzione `txt2bin` `srcTxtFile` è il nome di un file da aprire in modalità lettura tradotta (testo), mentre `dstBinFile` è il nome di un file da aprire in modalità scrittura non tradotta (binario).
Per la funzione `bin2txt` `srcBinFile` è il nome di un file da aprire in modalità lettura non tradotta (binario), mentre `dstTxtFile` è il nome di un file da aprire in modalità scrittura tradotta (testo).

Il file `srcTxtFile` passato alla funzione `txt2bin` contiene un numero intero codificabile in una variabile di tipo `char`, seguito da un numero intero codificabile in una variabile di tipo `int` e un numero a virgola mobile codificabile in una variabile di tipo `float` (tutti separati da spazi). Un esempio del contenuto del file è:

```
54 -3000 12.4543
```

La funzione `txt2bin` deve leggere i 3 numeri e scriverli in sequenza sul file destinazione `dstBinFile`.
La funzione scriverà quindi 9 byte in tutto (1 per il `char`, 4 per l'`int` e 4 per il `float`).
La funzione `bin2txt` deve leggere il contenuto del file scritto dalla funzione `txt2bin` e scriverlo sul file `dstTxtFile` in formato testo in base 10, separando ogni numero con uno spazio.
Alla fine dell'esecuzione del programma, Il contenuto dei file `srcTxtFile` e `dstTxtFile` sarà lo stesso (a meno di eventuali differenze di arrotondamento del numero con la virgola).

La funzione ritorna 1 se riesce ad eseguire tutte le seguenti operazioni: aprire il file `srcTxtFile`, leggere tutti i numeri, chiudere il file, aprire il file `dstBinFile`, scrivere tutti i numeri, chiudere il file, aprire il file `srcBinFile`, leggere tutti i numeri, chiudere il file, aprire il file `dstTxtFile`, scrivere tutti i numeri, chiudere il file. Altrimenti ritorna 0.

In questo esercizio si usino come file sorgenti i file allegati all'esercitazione:

file_sorgente_1.txt
file_sorgente_2.txt
file_sorgente_3_sbagliato.txt

Esercizio 2

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_copy(const struct matrix *mat);
```

La struct consente di rappresentare matrici di dimensioni arbitraria, dove `N` è il numero di righe, `M` 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 ad una matrice e deve ritornare una nuova matrice, allocata dinamicamente sull'heap, con le stesse dimensioni e con lo stesso contenuto, ovvero una copia della matrice. Il puntatore `mat` non sarà mai `NULL`.

Esercizio 3

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_leggi(const char *filename);
```

la funzione `mat_leggi()` accetta come parametro una stringa C contenente il nome di un file di testo da aprire in modalità tradotta dal quale leggere una `struc matrix`.
Nel file una matrix viene rappresentata con il seguente formato: prima ci sono due numeri interi in formato testo in base 10 separati da whitespace che rappresentano rispettivamente il numero di righe e di colonne, segue poi il contenuto della matrice sotto forma di numeri con la virgola, sempre separati da whitespace.
Ad esempio, per rappresentare una matrice 2x3, potremmo avere:

```
2 3 0.3 3. 2.54 2.6e-3 1.114e3 5.87
```

Il 2 rappresenta il numero di righe, il 3 il numero di colonne e poi seguono i sei numeri con la virgola che fanno parte della matrice.

La funzione deve allocare dinamicamente la memoria per la nuova matrix letta dal file e restituire un puntatore ad essa. Se la lettura fallisce la funzione deve ritornare `NULL` (ricordatevi di liberare le risorse in caso di fallimento).

Esercizio 4

Nel file `main.c` è definita la seguente funzione main:

```
int main(void)
{
	size_t lung;
	char str[100] = "Questa e' una stringa lunga 41 caratteri.";

	riempi_lung(str, &lung);
}
```

Aggiungere nel file (sopra a questa funzione) gli include necessari e la definizione della funzione `riempi_lung` che mette nella variabile `lung` il numero di caratteri della stringa `str`. La funzione deve essere fatta in modo da non modificare in alcun modo la funzione `main()` fornita.