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)

In matematica, il minimo comune multiplo di due numeri interi a e b, indicato con $\mathrm{mcm}(a,b)$ è il più piccolo intero positivo multiplo sia di a che di b. Se $a = 0$ o $b = 0$ allora si ha che il minimo comune multiplo è zero.
Se a e b non sono entrambi nulli il minimo comune multiplo può essere calcolato usando il massimo comun divisore di a e b e la formula seguente:

$$\mathrm{mcm}(a,b) = \left|\frac{a}{\mathrm{MCD}(a,b)} \cdot b\right|$$

Nel file `mcm.c` implementare la definizione della funzione:

```
extern size_t mcm(int a, int b);
```

La funzione prende come input i due interi a e b e deve ritornare il loro minimo comune multiplo.

Ricordo che il massimo comune divisore di due numeri interi a e b, che non siano entrambi uguali a zero, è il più grande numero naturale di cui sia a sia b sono multipli. Per questa definizione, $\mathrm{MCD}(a,b) = \mathrm{MCD}(|a|,|b|)$.

Esercizio 2 (6 punti)

Nel file `cornice.c` implementare la definizione della funzione:

```
extern void stampa_cornice(const char *s);
```

La funzione deve inviare a `stdout` la stringa passata come parametro circondandola con una cornicetta composta dei caratteri `/` e `\` agli spigoli superiori, di `` ` `` (apice inverso, carattere 96 della tabella ASCII) e `'` (apice normale) agli spigoli inferiori e di `-` e `|` sui lati. Prima e dopo la stringa bisogna inserire uno spazio. Ad esempio chiamando la funzione con `s = "ciao"`, la funzione deve inviare su `stdout`:

```
/------\↵
| ciao |↵
`------'↵
```

Ovvero (visualizzando ogni carattere in una cella della seguente tabella):

/	-	-	-	-	-	-	\	<a capo>
|	<sp.>	c	i	a	o	<sp.>	|	<a capo>
`	-	-	-	-	-	-	'	<a capo>

Gli <a capo> a fine riga sono obbligatori per una soluzione corretta.

Esercizio 3 (7 punti)

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 double *bordo_esterno(const struct matrix *m, size_t *new_size);
```

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 ad una matrice e un puntatore a una variabile di tipo `size_t` e deve ritornare un puntatore a un vettore di double allocato dinamicamente sull'heap contenente gli elementi che costituiscono il bordo esterno della matrice in senso orario a partire dal primo elemento in alto a sinistra. La funzione deve aggiornare il valore della variabile puntata da `new_size` con il numero di elementi che costituiscono il vettore.

Ad esempio, se consideriamo la seguente matrice:

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

La funzione dovrà ritornare un puntatore a un vettore contenente i valori `{ 1.0, 2.0, 3.0, 6.0, 9.0, 8.0, 7.0, 4.0 }` e aggiornare il valore della variabile puntata da `new_size` a 8.
Se la matrice passata contiene un solo elemento il vettore ritornato deve contenere solamente quell'elemento.
Se il puntatore alla matrice passato alla funzione è `NULL` la funzione ritorna `NULL`.

Esercizio 4 (7 punti)

Creare i file `dictionary.h` e `dictionary.c` che consentano di utilizzare le seguenti strutture:

```
struct pair {
	char key[256];
	char value[256];
};

struct dictionary {
	size_t size;
	struct pair *data;
};
```

e la funzione:

```
extern char *find(const struct dictionary *dict, const char *key);
```

La struttura `pair` permette di memorizzare una coppia chiave-valore dove entrambi gli elementi sono costituiti da stringhe di al massimo 256 caratteri (terminatore compreso). La struttura `dictionary` invece permette di memorizzare un vettore di `pair`.

La funzione accetta in input un puntatore a una `struct dictionary` e una stringa C e deve cercare nel dizionario una `pair` avente come chiave una stringa uguale a quella passata nel parametro `key` e ritornare il puntatore all'elemento trovato.

La funzione ritorna `NULL` se almeno uno dei due parametri è `NULL` o se non viene trovato nessun elemento che ha la chiave cercata. Tutte le chiavi del dizionario sono uniche.

Esercizio 5 (8 punti)

Creare i file `leggi_linee.h` e `leggi_linee.c` che consentano di utilizzare la seguente funzione:

```
extern char **leggi_linee(const char *filename);
```

La funzione accetta in input una stringa C che contiene il nome di un file da aprire in modalità lettura tradotta. La funzione deve ritornare un vettore di stringhe C contenente tutte le linee del file.
Il doppio puntatore ritornato dalla funzione dovrà puntare alla prima stringa (la prima linea del file). Per indicare il termine del vettore ritornato dalla funzione non si usa un'altra variabile per memorizzare la dimensione ma, similmente a come viene fatto per le stringhe C, dopo il suo ultimo elemento viene allocato spazio anche per un puntatore NULL.

Consideriamo il seguente file di esempio:

```
Prima riga↵
Seconda riga↵
Terza riga↵
```

In questo caso la funzione dovrebbe ritornare un doppio puntatore a char che punterà al primo di quattro puntatori singoli a char i quali a loro volta punteranno ai primi caratteri di tre stringhe C,

rispettivamente “Prima riga”, “Seconda riga” e “Terza riga”, mentre il quarto sarà un puntatore `NULL`.
Le righe lette NON dovranno contenere il carattere “a capo” alla fine e si può assumere che le linee siano lunghe al massimo 255 caratteri, carattere “a capo” compreso. Se alla funzione viene passato un puntatore `NULL` o se non è possibile aprire il file la funzione dovrà ritornare `NULL`. Se invece il file esiste ma non contiene alcun carattere, la funzione ritornerà un puntatore ad un puntatore NULL.