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 una funzione main() che ne testi il funzionamento. ***Fate progetti diversi per ogni esercizio.***

Esercizio 1

Creare un file `main.c`. Nel file, si realizzi in linguaggio C la funzione corrispondente alla seguente dichiarazione:

```
extern size_t lungh (const char *str);
```

La funzione riceve un puntatore a un array di char zero terminato (stringa C) e restituisce il numero di caratteri contenuti (senza il terminatore). Un esempio di chiamata è il seguente:

```
int main (void) {
	char s[] = "Ecco la stringa di prova";
	size_t len;

	len = lungh(s);
}
```

In questo caso `len` vale 24.

Esercizio 2

Creare un file `main.c`. Nel file, si scriva:

```
01	#include <stdlib.h>
02
03	int main(void)
04	{
05		double a[] = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 };
06		size_t i, n = sizeof a / sizeof a[0];
07
08		for (i = 0; i<n; ++i) {
09			double d = a[i];
10			potenza(&d,i);
11			a[i] = d;
12		}
13	}
```

Aggiungere sopra al main la definizione della funzione `potenza()` che eleva `d` alla i e mette il risultato di nuovo in d. La funzione deve essere fatta in modo da non modificare il main fornito.

Esercizio 3

Una volta risolto l'esercizio precedente, senza modificare la funzione `potenza()`, eliminare le righe 09 e 11. Modificare quindi la riga 10 in modo che il programma produca lo stesso risultato. Bonus: nella riga 10 non utilizzare l'operatore &.

Esercizio 4

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

```
unsigned int conta_spazi (const char* s);
```

La funzione accetta come parametro un puntatore a un array di char zero terminato (stringa C) e deve restituire il numero di caratteri <spazio> presenti nella stessa.
Ad esempio, data la stringa “prova stringa in cui contare gli spazi” la funzione deve ritornare il valore 6.

Esercizio 5

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

```
extern int cerca_primo (const char *s, char c);
```

La funzione accetta come parametro un puntatore a un array di char zero terminato (stringa C) in cui cercare la prima occorrenza del carattere c. La funzione restituisce l’indice di c nella stringa s (come al solito, partendo da 0 per il primo carattere). Nel caso il carattere non sia presente deve ritornare -1. Ad esempio se cercassimo in `s=“aereo”` il carattere `c='e'` dovrebbe ritornare 1; se cercassimo il carattere `c='x'` dovrebbe ritornare -1.

Esercizio 6

Sia data la struct seguente:

```
struct punto {
	double x, y;
};
```

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

```
extern int colineari(struct punto p1, struct punto p2, struct punto p3);
```

La funzione ritorna 1 se p_1, p_2 e p_3 giacciono sulla stessa retta, altrimenti 0. L’equazione da verificare è la seguente:

$$(x_3 - x_2)(y_1 - y_2) = (y_3 - y_2)(x_1 - x_2)$$

Esercizio 7

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

```
struct complesso {
	double re,im;
};
```

e la funzione:

```
extern void prodotto_complesso (struct complesso *comp1, const struct complesso *comp2);
```

La funzione `prodotto_complesso` esegue il prodotto dei due valori `comp1` e `comp2` e mette il risultato in `comp1`. Si ricorda che il prodotto di numeri complessi si esegue così:

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc)$$