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

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

La funzione accetta come parametro un array di char zero terminato (stringa C) in cui cercare l'ultima occorrenza del carattere c, ovvero la prima partendo dalla fine. 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 3; se cercassimo il carattere `c='x'` dovrebbe ritornare -1.

Esercizio 2

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

```
extern double ln (double x);
```

La funzione deve calcolare il valore di $\ln x$, ovvero del logaritmo naturale, utilizzando la seguente equazione:

$$\ln x = 2\sum_{n=0}^{\infty} \frac{1}{2n+1}\left(\frac{x-1}{x+1}\right)^{2n+1}$$

Nella formula l'iterazione prosegue fino all'infinito, ma nella pratica potete fare un numero limitato di iterazioni. Iniziate con 10 iterazioni eseguendo il debug e capendo che cosa succede. Poi trovate una soluzione che termini quando il risultato non cambia rispetto all'iterazione precedente, in modo simile a quanto visto a lezione per l'algoritmo della radice quadrata.

Trovare poi una soluzione che non utilizzi alcuna funzione aggiuntiva, ma che riesca a fare nel ciclo stesso il calcolo di $\left(\frac{x-1}{x+1}\right)^{2n+1}$.

Esercizio 3

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

```
struct vector {
	size_t size;
	double *data;
};
```

e le funzioni:

```
extern void vector_create(struct vector *v, size_t length);
extern void vector_destroy(struct vector *v);

extern struct vector *new_vector(unsigned int length);
extern void delete_vector(struct vector *v);

extern int vector_set(struct vector *v, size_t pos, double value);
extern int vector_get(const struct vector *v, size_t pos, double *pvalue);
```

`vector_create` serve a inizializzare i dati della struttura puntata da `v`, impostando `size` a `length` e allocando `length` double, mettendo poi il puntatore in `data`;
`vector_destroy` imposta `size` a 0, dealloca la memoria puntata da `data` e imposta `data` a `NULL`;

`new_vector` alloca una `struct vector`, la inizializza impostando `size` a `length` e allocando `length` double, mettendo poi il puntatore in `data` e ritorna un puntatore alla struttura così creata.
`delete_vector` dealloca la memoria puntata da `data` e da `v`.

`vector_set`, se `pos` è minore di `size`, imposta l'elemento di posizione `pos` dell'array al valore `value` e ritorna 1, altrimenti ritorna 0.
`vector_get`, se `pos` è minore di `size`, imposta il valore puntato da `pvalue` al valore dell'elemento di posizione `pos` dell'array e ritorna 1, altrimenti ritorna 0.

Esercizio 4

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

```
extern char *concatena(const char *prima, const char *seconda);
```

La funzione riceve due puntatori a stringhe di caratteri zero terminate e alloca dinamicamente sull'heap sufficiente memoria per contenerle entrambe (incluso un terminatore) e copia nel nuovo spazio allocato la prima, seguita dalla seconda. Un esempio di chiamata è il seguente:

```
int main(void) {
	char s1[] = "Questa e' la ";
	char s2[] = "stringa risultante.";
	char *s;

	s = concatena(s1, s2);

	free(s);
}
```

In questo caso s punterà ad un vettore di caratteri contenente "Questa e' la stringa risultante.". Se uno dei puntatori (`prima` o `seconda`) è NULL o punta ad una stringa vuota (cioè il primo carattere vale 0), la funzione lo tratta come una stringa di lunghezza 0. Ad esempio chiamando `concatena("a","")`, si allocano 2 char e la stringa ritornata contiene il carattere `'a'` e il carattere 0.

Esercizio 5

Creare il file `encrypt.c` che contenga la definizione della seguente funzione:

```
extern void encrypt(char *s, size_t n);
```

La funzione accetta una sequenza `s` di `n` char e la codifica sostituendo ad ogni char il suo valore trasformato con uno XOR bit a bit con il valore esadecimale AA. Per le proprietà dello XOR, l'operazione è invertibile, quindi riapplicando la funzione sulla sequenza codificata si riottiene quella originale.