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.***

Attenzione! La Microsoft ha definito funzioni non standard più sicure di quelle definite nello standard e segnala l'uso di funzioni considerate pericolose con questo warning:

```
warning C4996: '<nome funzione>': This function or variable may be unsafe. Consider using
<nome funzione sicura> instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online
help for details.
```

In questo corso utilizzeremo solo le versioni standard e quindi per convincere Visual Studio a fare il suo dovere aggiungete **prima di ogni #include** la seguente definizione

```
#define _CRT_SECURE_NO_WARNINGS
```

Esercizio 1

Creare il file conversione.c in cui deve essere definita la funzione corrispondente alla seguente dichiarazione:

```
extern char *converti(unsigned int n);
```

La funzione accetta come parametro un numero naturale o nullo n e deve restituire un puntatore ad un array di caratteri zero terminato allocato dinamicamente, contenente la rappresentazione in base 10 del numero intero, con le singole cifre rappresentate in ASCII. Ad esempio: se la funzione riceve in input n=4355, alloca 5 byte e li riempie con "4355", ovvero { 52, 51, 53, 53, 0 }, o anche { 0x34, 0x33, 0x35, 0x35, 0x00 }.

Esercizio 2

Creare i file `lettura.h` e `lettura.c` che consentano di utilizzare la seguente macro:

```
#define FILE_NOT_FOUND -2
```

e la funzione:

```
extern int leggiprimo(const char *filename);
```

La funzione riceve in input il nome di un file come stringa C e deve: aprire il file in lettura in modalità tradotta (testo), se l'apertura fallisce ritornare FILE_NOT_FOUND, altrimenti leggere il primo carattere dal file, se la lettura fallisce ritornare EOF, altrimenti ritornare il carattere letto. Ricordarsi comunque di chiudere il file se l'apertura è andata a buon fine.

Esercizio 3

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

```
extern int scrivimaiuscolo(const char *filename);
```

La funzione riceve in input il nome di un file come stringa C e deve: aprire il file in lettura in modalità tradotta (testo), se l'apertura fallisce ritornare 0, altrimenti leggere tutti i caratteri dal file e scriverli su `stdout` in maiuscolo (ovviamente solo se sono lettere minuscole), infine ritornare 1. Ricordarsi di chiudere il file se l'apertura è andata a buon fine.

Esercizio 4

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 int vector_add_to(struct vector *dst, const struct vector *src);
extern struct vector *vector_add(const struct vector *src1, const struct vector *src2);
```

`vector_add_to` verifica se `dst` e `src` sono grandi uguali e se no ritorna 0. In caso contrario, somma a ogni elemento di `dst` il corrispondente elemento di `src` e ritorna 1;
`vector_add` verifica se `src1` e `src2` sono grandi uguali e se no ritorna `NULL`. In caso contrario, crea un nuovo vettore grande come i precedenti e mette in ogni elemento la somma dei corrispondenti elementi di `src1` e `src2` e ritorna 1.