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)

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

```
extern double *probabilita(const uint32_t *v, size_t n);
```

La funzione riceve in input un puntatore a un vettore che contiene `n` elementi di tipo `uint32_t`, dove l'elemento di indice `i` rappresenta la frequenza $F(i)$ con cui il valore i compare in una popolazione.

La funzione deve restituire un nuovo vettore allocato dinamicamente su heap, in cui l'elemento di indice `i` è la probabilità $P(i)$ del valore i, calcolata come:

$$P(i) = \frac{F(i)}{\sum_{k=0}^{n-1} F(k)}$$

Se `v` è `NULL`, `n=0`, o tutti gli elementi di `v` sono uguali a zero, la funzione restituisce `NULL`.

Esercizio 2 (6 punti)

Nel file `inversione.c` implementare la definizione della seguente funzione:

```
extern bool inverti_case(const char* nomefilein, const char* nomefileout);
```

La funzione accetta come parametro un nome di file di input (stringa C zero terminata) da aprire in modalità lettura tradotta (testo) e un nome di file di output (stringa C zero terminata) da aprire in modalità scrittura tradotta (testo). La funzione deve produrre un file di output con tutti i caratteri di quello di input, ma se i caratteri sono minuscoli deve renderli maiuscoli e viceversa.

Ad esempio, dato in input il file:

```
abc123ABC!!!
```

in output produrrà

```
ABC123abc!!!
```

La funzione ritorna `false` se non riesce ad aprire il file di input o di output.

Esercizio 3 (7 punti)

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

```
extern void stampa_a(uint8_t n);
```

La funzione stampa su `stdout` una lettera `A` maiuscola, costruita nel seguente modo:

- il lato obliquo sinistro è composto da `n` caratteri `/`.
- il lato obliquo destro è composto da `n` caratteri `\`.
- la linea orizzontale attraversa la riga centrale della figura ed è costituita da caratteri `-`.

Ad esempio, invocando la funzione con `n=5`, essa deve stampare

```
    /\
   /  \
  /----\
 /      \
/        \
```

Se `n` è pari o minore di `3`, la funzione non stampa nulla.

Esercizio 4 (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 struct matrix *mat_delete_col(const struct matrix *m, size_t i);
```

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 parametri un puntatore ad una matrice `m` e un indice di colonna `i` e deve restituire un puntatore a una nuova matrice allocata dinamicamente. La matrice è ottenuta copiando i dati di quella di input ad eccezione della colonna i-esima, con `i=0` per la prima colonna. Se `i` specifica una colonna non valida, se `m` ha una sola colonna, o se `m` è `NULL`, la funzione restituisce `NULL`.

Ad esempio, data la matrice

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

la funzione, chiamata con il parametro `i` uguale a 1, restituisce la nuova matrice

$$\begin{pmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \end{pmatrix}$$

Esercizio 5 (8 punti)

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

```
extern char *somma_stringhe(const char *a, const char *b);
```

La funzione riceve in input due puntatori a stringhe C `a` e `b`, ciascuna delle quali contiene la rappresentazione ASCII di un numero naturale composto da una quantità variabile di cifre.

La funzione deve restituire un puntatore ad una nuova stringa C allocata dinamicamente nella memoria heap, che rappresenta la somma di `a` e `b`. Il risultato non deve contenere zeri superflui prima della cifra più significativa.

Ad esempio, date le stringhe

```
a = "12345"
b = "111"
```

La funzione deve restituire la stringa

```
"12456"
```

Se `a` e `b` rappresentano entrambe il numero `"0"`, la funzione restituisce `"0"`.

Se `a` o `b` valgono `NULL`, la funzione restituisce `NULL`.

Le stringhe di input saranno sempre correttamente formattate.