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)

Creare i file `bytearray.h` e `bytearray.c` che contengano rispettivamente dichiarazione e definizione della seguente funzione (il tipo `uint8_t` è definito nell'header `<stdint.h>`)

```
extern uint8_t *absdiff(const uint8_t *a, const uint8_t *b, size_t n);
```

La funzione accetta come parametri due puntatori a vettori di byte e il numero di elementi di ogni vettore. La funzione deve ritornare un puntatore ad un area di memoria, allocata dinamicamente sull'heap, contenente il valore assoluto della differenza tra gli elementi di posizione corrispondente nei vettori `a` e `b`. Ad esempio, con $a = [7,3,1]$ e $b = [4,3,2]$, la funzione dovrebbe ritornare $[3,0,1]$. Se `n` è 0, i puntatori `a` e `b` non devono essere utilizzati e bisogna ritornare `NULL`.

Esercizio 2 (punti 6)

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_creatediag(const double *values, size_t n);
```

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 a un vettore di valori e una dimensione `n` e deve ritornare una matrice diagonale di lato `n`, allocata dinamicamente sull'heap, la cui diagonale principale contiene i valori del vettore in ingresso. Se `n` è 0, il puntatore `values` non deve essere utilizzato e bisogna ritornare una matrice con `rows=cols=0` e `data` a qualsiasi valore che possa essere poi passato ad una `free()`, ad esempio `NULL` o `malloc(0)`.

Ad esempio, utilizzando il vettore { 1, 2, 3 }, `mat_creatediag()` ritorna la matrice:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

Esercizio 3 (7 punti)

L'alfabeto farfallino è un gioco linguistico per bambini che consiste nel raddoppiare ogni vocale con l'aggiunta di una f interposta: per esempio, "a" diventa "afa", "e" diventa "efe", e così via (quindi "ciao" diventa "cifiafaofo").
Nel file `farfallino.c` implementare la definizione della seguente funzione:

```
extern char *farfallino_decode(const char *s);
```

La funzione accetta una stringa C (zero terminata) codificata con l'Alfabeto Farfallino e ritorna una stringa C allocata dinamicamente sull'heap, contenente la stringa decodificata. Ad esempio passando alla funzione la stringa "afaifiufuofolafa" deve ritornare "aiuola".

Il puntatore ritornato deve puntare ad un'area di memoria allocata dinamicamente sull'heap **grande esattamente il numero di byte necessari a contenere la stringa risultante** e se `s` è NULL deve ritornare NULL.

Esercizio 4 (7 punti)

La trasformata coseno discreta (DCT) utilizzata nel JPEG è basata sulla seguente definizione:

$$S(u) = \alpha(u) \sum_{x=0}^{N-1} s(x) \cos \frac{(2x+1)u\pi}{2N}$$

con $u \in [0, N-1]$. Dato quindi un vettore N-dimensionale s produce in uscita un altro vettore N-dimensionale trasformato S. Ogni elemento di S viene calcolato utilizzando tutti gli elementi del vettore s, moltiplicati per diversi valori a seconda di u e di x. $\alpha(u)$ è definito nel seguente modo

$$\alpha(u) = \begin{cases} \sqrt{1/N} & u = 0 \\ \sqrt{2/N} & u \neq 0 \end{cases}$$

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

```
extern void dct(const double *in, double *out, size_t N);
```

La funzione deve riempire il vettore all'indirizzo puntato da `out` con gli `N` coefficienti calcolati a partire dal vettore puntato da `in`, secondo la formula precedente.
Ad esempio se $s = \{3, 5\}$, $S(0) = \sqrt{1/2}\left(3 \cdot \cos\frac{(2\cdot0+1)\cdot0\cdot\pi}{2\cdot2} + 5 \cdot \cos\frac{(2\cdot1+1)\cdot0\cdot\pi}{2\cdot2}\right) = 5{,}657$ e $S(1) = \sqrt{2/2}\left(3 \cdot \cos\frac{(2\cdot0+1)\cdot1\cdot\pi}{2\cdot2} + 5 \cdot \cos\frac{(2\cdot1+1)\cdot1\cdot\pi}{2\cdot2}\right) = \left(3 \cdot \cos\frac{1}{4}\pi + 5 \cdot \cos\frac{3}{4}\pi\right) = -1{,}414$

Esercizio 5 (8 punti)

Un file di testo per la configurazione di un programma software contiene sequenze di coppie (parola chiave, valore), una per ogni riga, che rispettano questa sintassi:

1) zero o più spazi o tab
2) una sequenza di caratteri senza spazi o tab
3) uno o più spazi o tab
4) il carattere '='
5) uno o più spazi o tab
6) una sequenza di caratteri senza spazi o tab
7) zero o più spazi o tab e il carattere <a capo>

Le righe che non rispettano questa sintassi o che iniziano con il simbolo '#' vanno saltate.

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

```
struct keyval {
    char k[256];
    char v[256];
};
```

e la funzione:

```
extern struct keyval *read_config(const char *filename, size_t *n);
```

La funzione deve aprire il file (ricordandosi di chiuderlo alla fine) leggerne ogni riga, verificare che la riga rispetti la sintassi prevista e in quel caso aggiungere un elemento ad un vettore di `struct keyval`, allocato dinamicamente sull'heap. Al termine, la funzione restituisce il puntatore al vettore così allocato e imposta nella variabile puntata da `n` il numero di elementi letti correttamente.

Ad esempio il file seguente:

```
primo = uno
 secondo = due
    terzo =     tre
# quarto=
quinto =
#sesto = 123
settimo = 456
ottavo = due cose
```

contiene 4 righe corrette: primo, secondo, terzo e settimo. Infatti quarto e sesto cominciano con #, quinto non ha un valore dopo l'uguale e ottavo ha dopo l'uguale una sequenza che contiene lo spazio.

Suggerimento: svolgere questo esercizio per passi. Prima fate una funzione in grado di leggere tutte le righe, poi una che ritorni anche il vettore allocato correttamente con k e v uguali all'intera riga letta, infine occupatevi di spezzare la riga all'uguale.