Descrizione del programma

Scrivere un programma in C che:

- **A** prenda in input da tastiera un intero positivo “n”;
- **B** legga un file di testo "input.txt". Si assuma che il file di testo contenga sequenze di parole separate da spazi. Si assuma una lunghezza massima per ciascuna parola pari di 100 caratteri, mentre una lunghezza massima di ciascuna riga pari a 1000 caratteri. Il programma dovrà creare una lista di struct a partire dal contenuto del file. Ogni struct deve rappresentare una riga del file di input e deve contenere due campi: una stringa "word” e un intero "count". La stringa "word" deve contenere la prima parola nella riga del file di input, mentre "count" deve contenere il numero di occorrenze di tale parola nella riga;
- **C** una volta creata la lista, elimini da essa le parole aventi un numero di occorrenze inferiore a “n”;
- **D** stampi a schermo le parole presenti nella lista con il relativo numero di occorrenze.

Specifiche

Il programma potrà essere articolato in un unico file sorgente, ma dovrà contenere almeno le seguenti funzioni con opportuni parametri formali:

- **readN**: funzione che permette di leggere l’intero n come parametro passato da riga di comando. Questa funzione dovrà gestire correttamente eventuali errori di input, stampare messaggi di errore su standard error e terminare il programma opportunamente ove necessario;
- **readFile**: funzione che apre il file di input (il cui nome è passato mediante un apposito parametro formale), legge le righe e restituisce la lista concatenata di struct;
- **filterList**: funzione che elimina dalla lista le parole con meno di “n” occorrenze;
- **printList**: funzione che stampa il contenuto della lista concatenata.

Note

Durata della prova: 120 minuti

Generazione di numeri pseudocasuali:

- Si consideri la seguente funzione get_random() per la generazione di numeri pseudo-casuali interi positivi (qualora necessaria):

```
// Scaricabile da: https://pastebin.com/f6eAKNQy
unsigned int get_random() {
    static unsigned int m_w = 123456;
    static unsigned int m_z = 789123;
    m_z = 36969 * (m_z & 65535) + (m_z >> 16);
    m_w = 18000 * (m_w & 65535) + (m_w >> 16);
```

```
    return (m_z << 16) + m_w;
}
```

- NB: Ai fini della generazione di numeri in virgola mobile, si faccia uso della costante UINT_MAX (<limits.h>) unitamente alla funzione get_random().

È VIETATO usare variabili globali.

Output di controllo

Si consideri il seguente file “input.txt”:

```
frase questa frase è una bella frase
testo ma non è molto utile
albero il nostro albero non può essere un albero qualsiasi
eccomi qui qui sei tu eccomi eccomi eccomi
ancora tu ma non dovevamo ancora tu tu tu ancora?
ecco alto ello alto ecco
eccomi qui qui sei tu eccomi eccomi eccomi
```

L’esecuzione del programma con n=3 (`./main 3`) dovrà produrre il seguente output:

```
frase 3
albero 3
eccomi 4
eccomi 4
```