1 Esercizio 1

Scrivere nel file `esercizio1.cc` un programma che, dato un file di input contente delle parole, generi un secondo file in output che contenga le stesse parole, ma in ordine inverso rispetto al file iniziale. Per essere copiata, una parola deve essere composta da un numero di caratteri **pari**.

Il programma dovrà accettare due parametri da riga di comando: il nome del file in input e il nome del file su cui effettuare l'output, in questo ordine.

Il programma dovrà anche implementare dei controlli sul numero di argomenti passati da riga di comando e sull'apertura dei file (in caso di un file di input non esistente).

Procediamo ora con un esempio. Dato in input il file `input_A`, contenente le seguenti parole:

```
Impara a risolvere tutti i problemi che sono stati risolti.
Richard Feynman
```

se l'eseguibile è `a.out`, allora il comando

```
./a.out input_A output
```

genererà un file chiamato `output` che conterrà le seguenti parole:

```
risolti. sono problemi Impara
```

Note

- Per semplicità, considerate come una *parola* qualsiasi stringa di caratteri compresa tra due spazi bianchi (e/o separatori di tabulazione, nuova linea e fine file). Quindi, sono da considerarsi parole anche stringhe come `"andato:"` o `"!esempio"`.
- Il file in input può contenere al massimo 10000 parole e ognuna di queste parole è formata da al massimo 100 caratteri.
- Non è consentito l'utilizzo di librerie diverse da `<iostream>` ed `<fstream>` pena l'**annullamento dell'esercizio**.
- E' consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema.

2 Esercizio 2

Scrivere nel file `esercizio1.cc` un programma che, dato un file di input contente delle parole, generi un secondo file in output che contenga le stesse parole, ma in ordine inverso rispetto al file iniziale. Per essere copiata, una parola deve essere composta da un numero di caratteri **dispari**.

Il programma dovrà accettare due parametri da riga di comando: il nome del file in input e il nome del file su cui effettuare l'output, in questo ordine.

Il programma dovrà anche implementare dei controlli sul numero di argomenti passati da riga di comando e sull'apertura dei file (in caso di un file di input non esistente).

Procediamo ora con un esempio. Dato in input il file `input_A`, contenente le seguenti parole:

```
Impara a risolvere tutti i problemi che sono stati risolti.
Richard Feynman
```

se l'eseguibile è `a.out`, allora il comando

```
./a.out input_A output
```

genererà un file chiamato `output` che conterrà le seguenti parole:

```
Feynman Richard stati che i tutti risolvere a
```

Note

- Per semplicità, considerate come una *parola* qualsiasi stringa di caratteri compresa tra due spazi bianchi (e/o separatori di tabulazione, nuova linea e fine file). Quindi, sono da considerarsi parole anche stringhe come `"andato:"` o `"!esempio"`.
- Il file in input può contenere al massimo 10000 parole e ognuna di queste parole è formata da al massimo 100 caratteri.
- Non è consentito l'utilizzo di librerie diverse da `<iostream>` ed `<fstream>` pena l'**annullamento dell'esercizio**.
- E' consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema.

3 Esercizio 3

Scrivere nel file `esercizio1.cc` un programma che, dato un file di input contente delle parole, generi un secondo file in output che contenga le stesse parole, ma in ordine inverso rispetto al file iniziale. Per essere copiata, una parola deve essere composta da un numero di caratteri **maggiore di 4**.

Il programma dovrà accettare due parametri da riga di comando: il nome del file in input e il nome del file su cui effettuare l'output, in questo ordine.

Il programma dovrà anche implementare dei controlli sul numero di argomenti passati da riga di comando e sull'apertura dei file (in caso di un file di input non esistente).

Procediamo ora con un esempio. Dato in input il file `input_A`, contenente le seguenti parole:

```
Impara a risolvere tutti i problemi che sono stati risolti.
Richard Feynman
```

se l'eseguibile è `a.out`, allora il comando

```
./a.out input_A output
```

genererà un file chiamato `output` che conterrà le seguenti parole:

```
Feynman Richard risolti. stati problemi risolvere Impara
```

Note

- Per semplicità, considerate come una *parola* qualsiasi stringa di caratteri compresa tra due spazi bianchi (e/o separatori di tabulazione, nuova linea e fine file). Quindi, sono da considerarsi parole anche stringhe come `"andato:"` o `"!esempio"`.
- Il file in input può contenere al massimo 10000 parole e ognuna di queste parole è formata da al massimo 100 caratteri.
- Non è consentito l'utilizzo di librerie diverse da `<iostream>` ed `<fstream>` pena l'**annullamento dell'esercizio**.
- E' consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema.

4 Esercizio 4

Scrivere nel file `esercizio1.cc` un programma che, dato un file di input contente delle parole, generi un secondo file in output che contenga le stesse parole, ma in ordine inverso rispetto al file iniziale. Per essere copiata, una parola deve essere composta da un numero di caratteri **minore o uguale a 4**.

Il programma dovrà accettare due parametri da riga di comando: il nome del file in input e il nome del file su cui effettuare l'output, in questo ordine.

Il programma dovrà anche implementare dei controlli sul numero di argomenti passati da riga di comando e sull'apertura dei file (in caso di un file di input non esistente).

Procediamo ora con un esempio. Dato in input il file `input_A`, contenente le seguenti parole:

```
Impara a risolvere tutti i problemi che sono stati risolti.
Richard Feynman
```

se l'eseguibile è `a.out`, allora il comando

```
./a.out input_A output
```

genererà un file chiamato `output` che conterrà le seguenti parole:

```
sono che i a
```

Note

- Per semplicità, considerate come una *parola* qualsiasi stringa di caratteri compresa tra due spazi bianchi (e/o separatori di tabulazione, nuova linea e fine file). Quindi, sono da considerarsi parole anche stringhe come `"andato:"` o `"!esempio"`.
- Il file in input può contenere al massimo 10000 parole e ognuna di queste parole è formata da al massimo 100 caratteri.
- Non è consentito l'utilizzo di librerie diverse da `<iostream>` ed `<fstream>` pena l'**annullamento dell'esercizio**.
- E' consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema.