Laboratorio di Programmazione
(Corso di Laurea in Informatica)

ESAME del 14 Settembre 2017

Avvertenze

- VERRANNO CORRETTI SOLO E SOLTANTO I COMPITI IL CUI ESERCIZIO FILTRO FUNZIONA PERFETTAMENTE
- I programmi realizzati DEVONO rispettare le specifiche funzionali formite (input/output).
- Nello svolgimento dell'elaborato è possibile usare qualunque classe delle librerie standard di Java.
- Si raccomanda di salvare, compilare e fare upload delle soluzioni con una certa regolarità.
- Per la procedura di **consegna** si veda in fondo al documento.
- ATTENZIONE! Rispettare le specifiche dell'output! La correzione avviene in automatico mediante script che verificano la forma dell'output, aggiungere messaggistica ulteriore o utilizzare termini e/o spaziatura differenti significa invalidare il proprio lavoro.

Esercizio Filtro: *lunghezze di parole*

Descrizione

Scrivere una applicazione (una classe "LunghezzeParole" dotata del metodo `main`) che, dato un certo numero (non noto a priori) di parole su standard input (separate da "whitespace", cioè spazi, tab, enter, ecc.), visualizzi su standard output il numero massimo, medio (troncato all'intero) e minimo di caratteri presenti nelle parole in input (per il formato di output si vedano gli esempi più sotto).

Vincoli

Si assuma che l'input contenga almeno una riga di testo non vuota. L'input termina con *fine file* (i.e., in sistemi Linux la pressione simultanea dei tasti "Ctrl" e "d" su riga vuota se provate il programma da shell prompt). Il numero di parole presenti in input non è noto a priori e può essere disposto su più righe anch'esso non noto a priori.
Il file sorgente consegnato DEVE chiamarsi `LunghezzeParole.java` e contenere la classe **pubblica** `LunghezzeParole` con un `main` invocabile con: `$ java LunghezzeParole`

Esempi

Ad esempio se l'input fosse

```
abab    cdcd        efefefe

    ggg hihih mnomnomno
```

l'output dovrebbe essere il seguente:

```
max: 9
med: 5
min: 3
```

Se l'input invece fosse:

```
enea ecuba

        penelope


achille         briseide
```

l'output dovrebbe essere il seguente:

```
max: 8
med: 6
min: 4
```

---

1 Conteggio di numeri

1.1 Descrizione

Scrivere un programma (una classe "ContaParoleConNumeri" dotata del metodo `main`) che legge da standard input alcune stringhe contenenti o meno cifre. Le stringhe sono separate da caratteri di spaziatura (cioè newline, spazi, tab, ecc.). Il programma deve generare in output una statistica di quante stringhe sono state incontrate contenenti almeno una cifra e quante non contenevano alcuna cifra e del numero totale di cifre contenute nelle stringhe incontrate (si vedano esempi sotto). Il numero di stringhe e il numero delle righe non è noto a priori.

1.2 Vincoli

Non ci sono vincoli sul numero di stringhe e numeri su una riga, né sul numero di righe, né sul numero di caratteri di spaziatura (cioè newline, spazi, tab, ecc.) che li separano.
Il file sorgente consegnato DEVE chiamarsi `ContaParoleConNumeri.java` e contenere la classe **pubblica** `ContaParoleConNumeri`.

1.3 Esempi

Ad esempio, un input tipo:

```
1243        aaa   -3.5      bb      c54

    a.b     3-2     a-6


        plan9fromouterspace

1276767.7
    aslkdjaslk

    -92873981           76237263.989898
```

deve visualizzare il seguente output:

```
parole con cifre: 9
parole senza cifre: 4
numero totale di cifre: 42
```

---

2 Parole in ordine

2.1 Descrizione

Scrivere un programma (una classe "SortParole" dotata del metodo `main`) che legga da un file, il cui nome è passato per parametro sulla linea di comando, le parole in esso contenute e le visualizzi in ordine alfabetico, una per riga (se ci sono più occorrenze della stessa parola, verranno visualizzate tutte). Il numero di parole non è noto a priori e le parole sono separate da caratteri di spaziatura (newline, spazi, tab, ecc.).

2.2 Vincoli

NON implementate un vostro algoritmo di ordinamento.
Il file sorgente consegnato DEVE chiamarsi `SortParole.java` e contenere la classe **pubblica** `SortParole`.

2.3 Esempi

Ad esempio, se il file *testo* contiene:

```
nel mezzo del cammin di nostra vita


        mi ritrovai       per una selva        oscura
```

Eseguendo

```
$ java SortParole testo
```

si dovrà visualizzare:

```
cammin
del
di
mezzo
mi
nel
nostra
oscura
per
ritrovai
selva
una
vita
```

3 Conteggio di sottostringhe

3.1 Descrizione

Definire un metodo `int quanteVolteIn (String a, String b)` che calcola *ricorsivamente* il numero di occorrenze della stringa `a` nella stringa `b`.
Tale quantità può essere definita nel modo seguente:

- se la lunghezza di `a` supera quella di `b`, essa è uguale a 0
- altrimenti essa è uguale al numero di occorrenze di `a` nella stringa ottenuta da `b` escludendo il primo carattere, sommato a 1 oppure 0 a seconda che `a` sia/non sia "prefisso" di `b`.

Suggerimento: utilizzare i metodi `startsWith` e `substring` di `String`.

Scrivere un programma (una classe `QuanteVolteIn` dotata del metodo `main`) che, date due stringhe passate da linea di comando, stampi quante volte la prima è contenuta nella seconda sfruttando il metodo di cui sopra.

3.2 Vincoli

Il file sorgente consegnato DEVE chiamarsi `QuanteVolteIn.java` e contenere la classe **pubblica** `QuanteVolteIn`.

3.3 Esempi

```
$java QuanteVolteIn cdc  cdcdc
2
$

$java QuanteVolteIn cdc  cdcdccdcefdcdccdcdc
6
$
```

Consegna

Si ricorda che le classi devono essere tutte *public* e che vanno consegnati tutti (e soli) i file *.java* prodotti. NON vanno consegnati i *.class*. Per la consegna, eseguite l'upload dei SINGOLI file sorgente (NON un file archivio!) dalla pagina web: http://upload.di.unimi.it nella sessione che vi è stata indicata.

*** ATTENZIONE!!! ***

NON VERRANNO VALUTATI GLI ELABORATI CON ERRORI DI COMPILAZIONE O LE CONSEGNE CHE NON RISPETTANO LE SPECIFICHE (ad esempio consegnare un archivio zippato è sbagliato)
UN SINGOLO ERRORE DI COMPILAZIONE O DI PROCEDURA INVALIDA **TUTTO** L'ELABORATO.

Per ritirarsi fare l'upload di un file vuoto di nome `ritirato.txt`.
