Laboratorio di Programmazione
(Corso di Laurea in Informatica)

ESAME del 05 Luglio 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).
- I NOMI dei file DEVONO rispettare alla lettera le specifiche (compreso rispettare maiuscole e minuscole, singolari e plurali), pena l'invalidazione della prova
- 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 e non solo alla fine dell'esame.
- Allo scadere del tempo NON verranno più accettate consegne, si consiglia caldamente di caricare i file senza ridursi all'ultimo minuto!!!
- Si consiglia CALDAMENTE l'utilizzo dello script "checkNomiFile.sh" per verificare l'aderenza alla specifica sui nomi dei file e per fare un primo giro di compilazione
- 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: *PotenzeDiDue*

Descrizione

Un numero è una potenza di due se è pari a 1, o se si può ottenere moltiplicando successivamente 2 per se stesso.
Scrivere un programma (il file DEVE chiamarsi `PotenzeDiDue.java`) che, data una sequenza di interi positivi passata da *linea di comando*, emetta nel flusso d'uscita quanti tra essi sono potenze di due.

Vincoli

Nel caso (possibile) in cui la sequenza in input sia vuota si stampi semplicemente il messaggio "nessun input". Si assuma altrimenti che i dati in input siano nella forma attesa.

Esempi

Ecco un esempio di esecuzione

```
$java PotenzeDiDue 431 1 221 64 55 512
3
```

TrovaRigheColonneTutteZero

Descrizione

Scrivere un programma (il file DEVE chiamarsi `TrovaRigheColonneTutteZero.java`) che legga il contenuto di un file il cui nome è specificato da linea di comando. Si assuma (senza verificare) che il file contenga una tabella/matrice di caratteri '0' e '1'. Il numero di righe della tabella è arbitrario, il numero di colonne è arbitrario ma uguale per tutte le righe. Il programma deve produrre in output due serie (eventualmente vuote) di indici (1-based, cioè in cui l'indice della prima riga è 1) corrispondenti alle righe ed alle colonne formate da soli caratteri '0'.

Vincoli

Nel caso (possibile) in cui il nome specificato non corrisponda ad alcun file si stampi il messaggio "file non esistente". Si assuma altresì che i dati contenuti nel file siano nella forma attesa e che il file NON sia vuoto.

Esempi

Ecco un esempio di esecuzione. Si assuma che il file dati.txt si trovi nella directory di lavoro e che abbia il seguente contenuto

```
000000
010100
100010
000000
110000

$java TrovaRigheColonneTutteZero dati.txt
righe tutte zero:
1
4
colonne tutte zero:
3
6
```

DivideParoleMinMaiu

Descrizione

Scrivere un programma (il file DEVE chiamarsi `DivideParoleMinMaiu.java`) che, dato un file il cui nome è passato da *linea di comando*, emetta nel flusso d'uscita le parole che inziano con una lettera minuscola e quelle che iniziano con una maiuscola, opportunamente separate (come descritto nell'esempio sottostante). Si consideri come separatore (nel file di input) una qualsiasi sequenza di caratteri di "spaziatura" (cioè spazi, ritorni a capo, tabulazioni, ecc.). Le parole in output NON devono essere ordinate ed eventuali ripetizioni vanno riprodotte.

Vincoli

Per "parola" si intende una sequenza di soli caratteri alfabetici. Gli elementi (token) del file di input che non sono parole vanno semplicemente ignorati. Nel caso (possibile) in cui il nome specificato da linea di comando non corrisponda ad alcun file si stampi il messaggio "file non esistente".

Esempi

Ecco un esempio di esecuzione. Si assuma che il file testo.txt si trovi nella directory di lavoro e che abbia il seguente contenuto

```
Questo quello lui Lei
Io tu Noi Voi Marco78 paola1997

Essi loro
Nostro vostro mio Mio
```

```
$java DivideParoleMinMaiu testo.txt
iniziano con minuscole:
quello
lui
tu
loro
vostro
mio

iniziano con maiuscole:
Questo
Lei
Io
Noi
Voi
Essi
Nostro
Mio
```

Calcolatrice

Scrivere un programma (il file DEVE chiamarsi `Calcolatrice.java`) Calcolatrice che operi come una semplice calcolatrice in grado di memorizzare i risultati parziali. Il programma deve consentire all'utente di digitare attraverso il flusso di input standard espressioni nella forma seguente:

```
operatore numero
```

I seguenti operatori/comandi dovrebbero essere riconosciuti:
$+ - * / S E$
Il comando S indica al programma di memorizzare in un accumulatore il numero digitato, mentre E indica la fine dell'esecuzione. Le operazioni aritmetiche sono svolte sul contenuto dell'accumulatore e sul numero digitato, che funge da secondo argomento. Dopo ogni operazione il programma salva il risultato nell'accumulatore e dovrebbe produrre in output il contenuto dell'accumulatore. Il valore iniziale dell'accumulatore è 0.

Vincoli

Ogni operazione/comando viene specificata(o) su una riga diversa. Si assuma che il numero/tipo degli argomenti in una riga (separati da spazi) sia corretto. Eventuali argomenti aggiuntivi vanno ignorati. Qualora il primo argomento sia 'E', l'esecuzione termina. Eventuali anomalie causate dal mancato rispetto di queste assunzioni da parte dell'utente NON devono essere gestite.
Il programma deve invece controllare il caso della divisione per zero e la sintassi degli operatori (solo quelli specificati sono ammessi), producendo in entrambi i casi un opportuno messaggio in output, senza interrompere l'esecuzione e senza modificare l'accumulatore. I messaggi di errore sono "divisione per zero" e "operatore sconosciuto".

Esempi

Ecco un esempio di esecuzione:

```
$ java Calcolatrice
S 10
= 10.00
/ 2
= 5.00
- 55
= -50.00
S 100.25
= 100.25
* 4
= 401.00
E
$
```

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`.
